Dr. Dobb's Journal

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

#106 AUGUST 1985 $2.95 ($3.95 CANADA)

The Definitive C Compiler Review

Update on Small C

An Optimizer for C Code

An MSDOS "Make" Utility

CP/M Exchange Goes On Line

0 38351 16562

Software development isn't a mountainous task once you eliminate the high C errors.

When you can find and fix bugs at the earliest possible moment, creating software stops being such an uphill grind.

And the Smart/C Environment makes it possible. It's a complete, fully-integrated development environment for C that saves you from the creativity-inhibiting cycle of edit, compile, re-edit, re-compile, link, load, test, re-edit, re-compile, etc., ad infinitum. Smart/C puts the fun back in programming, because you spend your time creating... not waiting.

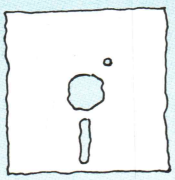
Here's why. Syntax errors are eliminated automatically as code is entered. Smart/C's highly integrated editor and interpreter allow you to interpret your program at any time in the creation process, so logic errors can be ferreted out as soon as the algorithm exists—long before any compile, link, or load.

The complete integration of the editor and interpreter means you can stop anywhere in the interpret cycle, edit, and then go right back into the interpreter exactly where you left off. Not only that, the screen-oriented user interface lets you see all operations, even interpretation, right on the listing of the code.

Circle **no. 4** on reader service card.

And to make maintenance programming easier, Smart/C's Migrator allows existing C code produced with any editor to be modified and run within the Smart/C Environment.

All of which makes Smart/C an excellent tool. It's flexible, non-restrictive, and lets you create elegant, readable, error-free programs that you can watch run with a great feeling of satisfaction.



Smart/C™

Free Demo Disk!

To fully appreciate Smart/C, you have to see it in action. For your free IBM PC MS-DOS demo disk, call us. Or write us on your company letterhead.
AGS Computers, Inc., Advanced Products Division, 1139 Spruce Drive, Mountainside, NJ 07092.
800-AGS-1313. In NJ, 201-654-4321.

Smart/C Features

The Smart/C Environment
- ☐ Fully integrated editor and interpreter
- ☐ Only one load brings them both in
- ☐ One command set
- ☐ Move between one another at will

Syntax Directed Editor
- ☐ vi-like command set
- ☐ Automatically provides formats for blocks, *for, case* and *if* statements

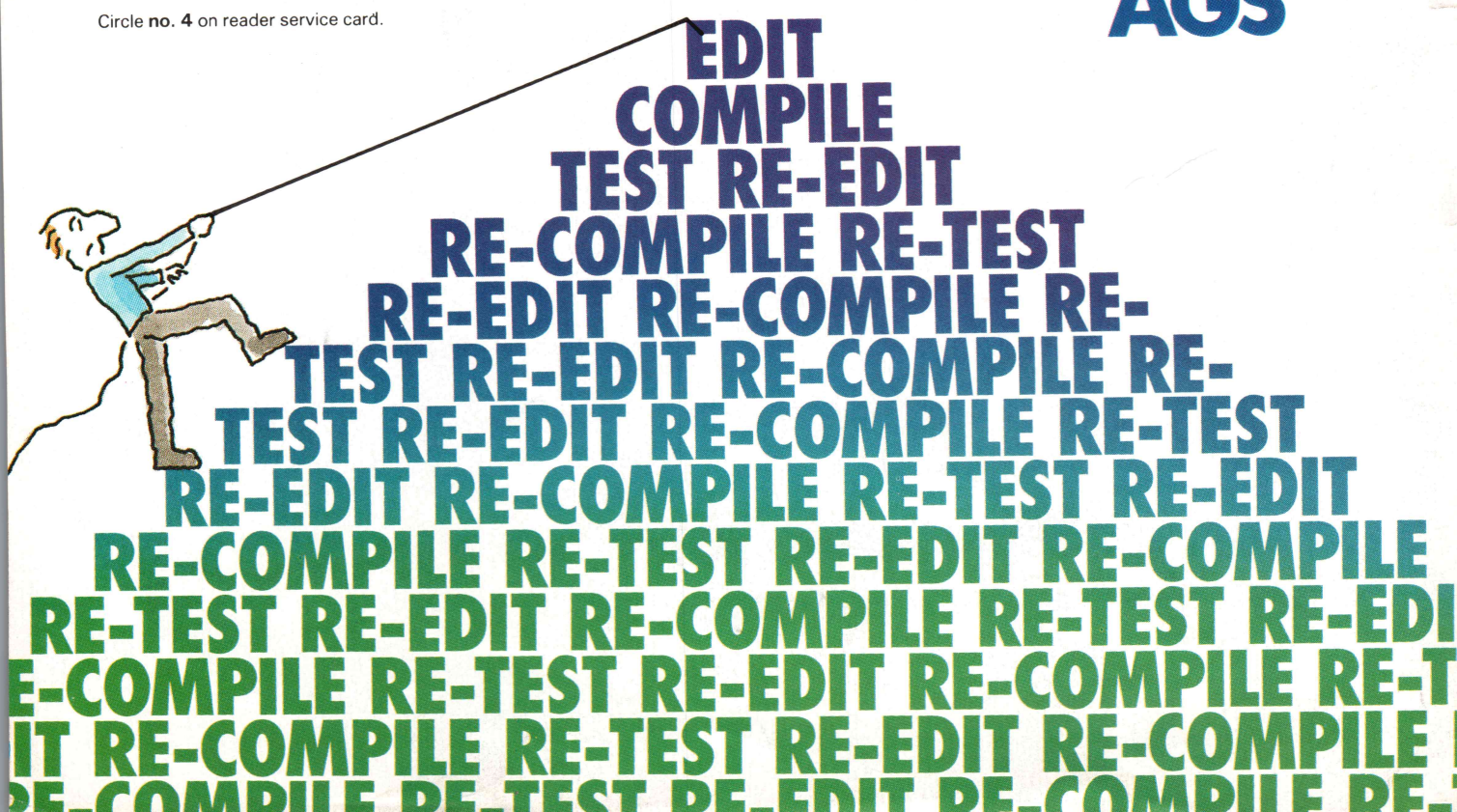
Interpreter
- ☐ Current module can call external modules during interpretation
- ☐ Has Include capability
- ☐ Totally precompilation—no incremental compile
- ☐ Can interpret partially defined files allowing for rapid prototyping
- ☐ Variable speed of interpretation
- ☐ Multiple windows with user-defined sizes

The Smart/C Migrator
- ☐ Allows C code produced with any editor to be interpreted by Smart/C
- ☐ Reformats for readability

Smart/C has been ported to UNIX™ System V Release 2, Berkeley 4.2, Xenix,™ and MS-DOS. Versions run on 8086- and 68000-based machines, as well as proprietary architectures. Smart/C runs on PCs, micros, supermicros, minis, and even mainframes.

Trademarks—Smart/C: AGS Computers, Inc.; UNIX: AT&T Bell Labs; Xenix and MS-DOS: Microsoft Corp; IBM PC: IBM Corp.

AGS



living C-personal™

GIVES LIFE TO C PROGRAMMING

Living C™ is the fully integrated interactive programming environment for C. By replacing the headaches of programming in C with total control and understanding – whether new to C or an expert – Living C is the exciting solution to maximize your creativity and productivity.

$99

The Living C editor is a true full function commercial editor, fully menu-driven with help facilities on call. The editor is fully integrated and can be used at all times throughout the design, development, maintenance and debugging of your C application.

FULL SCREEN EDITOR

Living C allows you to execute all your C source code on the screen. You control the code you wish to examine and the speed of animation. The cursor demonstrates exactly how your source is executing, enabling you to understand instantly how the application or prototype works – or why it doesn't!

ANIMATING C INTERPRETER™

Living C conforms to the full Kernighan and Ritchie standard. Living C not only highlights all errors discovered, but also offers comprehensive error diagnostics and useful hints to solve the problem. Corrections can be made immediately, using the fully integrated Living C editor.

FULL C SOURCE DEBUG

The Living C windows allow you to constantly monitor the variables and I/O of your applications even when you are "zooming" to your next breakpoint!

WINDOWS

I understand that I can write my applications in Living C, but what if I want to debug or modify an existing application – bearing in mind that 75% of my programming time is taken up with maintenance?

You simply compile in your C source and leave the rest to Living C! Living C not only enables you to understand how your application works but also ensures you can interpret a colleague's application and understand why it works or doesn't!! Once in Living C, the full suite of programming tools are automatically available. Combined with your rapid increase in productivity and understanding, you will now find that maintenance time is dramatically reduced.

So now I have written and tested my application, how can I use it?

You have 3 choices:

(i) Simply switch off the animation and use Living C as an interpreter
(ii) Recompile your application into your favourite C compiler (eg Microsoft, Lattice, Computer Innovations, Aztec)
(iii) Use the optional Living C code generator ($99)

What machines does Living C – Personal run on?

Living C – Personal is available for the IBM PC and all compatibles. You will need PCDOS, either twin floppy disk drives or a floppy and a hard disk with 192K RAM.

How do I order Living C – Personal?

Just fill out the coupon and send it to us along with $99 or call us on the toll free number **1 800 826-2612**

Circle **no. 122** on reader service card.

Living Software, 250 North Orange Avenue, Suite 820, Orlando, Florida 32801

Living C, Living C – Personal and Animating C Interpreter are trade marks of Living Software

Mail to: Living Software, 250 North Orange Avenue, Suite 820, Orlando, Florida 32801

To order your copy of Living C-Personal, please complete this form:

Please Send
Living C – Personal @ $99 x Quantity = $ _____
Code Generator @ $99 x _____ = $ _____
Handling & Shipping Orders = $ 10.00
Florida residents please include Florida Sales Tax = $ _____
TOTAL = $ _____

Payment Visa☐ MC☐ Check☐ Money Order☐
Credit Card expiry date _____/_____
Name on Card _____
Card# _____

Name _____
Shipping Address _____
City _____
State _____
Telephone _____ Zip _____

Living C – Personal is available for PC-DOS with min 192KB RAM.
COD's and Purchase Orders will not be accepted by Living Software.

Software Development Tools

C
Pascal
PLM
Assemblers
Simulators
VT-100 Emulator
CPM/80 Emulator

If:

Manufacturer-Compatible, Fully-Supported Cost-Effective, Cross and Native Development tools are important to you!

Then:

Let us tell you about MICROTEC RESEARCH's extensive line of field-proven software development tools for serious software developers.

Manufacturer Compatible

MICROTEC RESEARCH has been providing flexible and economical solutions for software developers since 1974. We've grown with the industry as our cross software development tools have transformed many large and small computers into powerful cross development systems for microprocessor applications. Our software tools are manufacturer compatible. Therefore, software made using manufacturers development systems may be more productively used in the MICROTEC RESEARCH cross-development environment.

Fully-Supported

MICROTEC RESEARCH's products are continually maintained and updated based upon the experience of thousands of installations worldwide. Revisions and modifications are distributed to licensees in warranty or covered by a service contract. Upgrades, which are major enhancements to a product's capabilities, are available at a significant discount. Our update/upgrade policy assures users they'll always be current with the fast moving world of microprocessor development.

Effective Differences

Beginning with product concept, through development, quality assurance, and post-sales support — **Quality, Compatibility,** and **Service** are the differences which set MICROTEC RESEARCH apart from the others.

Start Saving Time & Money

If you are a serious software developer, shopping for software development tools, call or write today for more information.
800-551-5554 In CA call **(408) 733-2919**

Target Microprocessors	Host Computers	Software Tools
8086/80186	DEC VAX, PDP-11	C Compilers
8096	DG MV-Series	Pascal Compilers
8080/8085	DG Eclipse	PLM Compiler
8051	Apollo	Assemblers
8048	UNIX Systems	Linking Loaders
Z8002	IBM PC	Librarians
Z80	Data General/ONE	Download
Z8	HP 150	Communications
NSC 800	DEC Rainbow	VT-100 Emulator
9900...	others	CPM/80 Emulator
68000/08/10		
6809		
6800/01/02		
6805		
6301		
6305		
6500		
G65SCXX, G65SC1XX		
R65C00, R6500/1...		
others		

for Serious Software Developers

MICROTEC is a trademark of Microtec Research, Inc. Santa Clara, CA

MICROTEC RESEARCH

3930 Freedom Circle, Suite 101, Santa Clara, CA 95054
Mailing Address: P.O. Box 60337, Sunnyvale, CA 94088
(408) 733-2919 • Telex (ITT) 4990808

Circle **no. 60** on reader service card.

Slash Programming Time in Half!

With FirsTime™

- Fast program entry through single keystroke statement generators.
- Fast editing through syntax oriented cursor movements.
- Dramatically reduced debugging time through immediate syntax checking.
- Fast development through unique programmer oriented features.
- Automatic program formatter.

FirsTime is a true syntax directed editor.

FirsTime ensures the integrity of your programs by performing all editing tasks like moves, inserts and deletes along the syntactic elements of a program. For example, when you move an IF statement, FirsTime will move the corresponding THEN and ELSE clauses with it.

Even FirsTime's cursor movements are by syntax elements instead of characters. The cursor automatically skips over blank spaces and required keywords and goes directly to the next editable position.

FirsTime is a Syntax Checker

FirsTime checks the syntax of your program statements, and also:

- Semantics like undefined variables and mismatched statement types.
- The contents of include files and macro expansions.
- Statements for errors as they are entered and warns you immediately.

FirsTime has Unique Features

No other editor offer these features:

The *Zoom command* gives you a top down view of your program logic.

The *View command* displays the contents of include files and macro expansions. This is valuable to sophisticated programmers writing complex code or to those updating unfamiliar programs.

FirsTime's *Transform command* lets you change a statement to another similar one with just two keystrokes. For example, you can instantly transform a FOR statement into a WHILE statement.

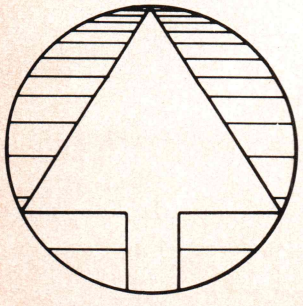
The *Move at Same Level command* moves the cursor up or down to the next statement at the same indentation level. This is very useful. For example, you can use it to locate the ELSE clause that corresponds to a given THEN clause or to traverse a program one procedure at a time.

FirsTime is a Program Formatter

FirsTime automatically indents statements as they are entered, saving you from having to track indentation levels and count spaces.

FirsTime is Unparalleled

FirsTime is the most advanced syntax directed editor available. It makes programming faster, easier and more fun.



TO ORDER CALL (201) 741-8188

or write:

Spruce Technology Corporation

189 E. Bergen Place
Red Bank, NJ 07701

Circle **no. 65** on reader service card.

In Germany, Austria and Switzerland contact:

Markt & Technik Software Verlag
Munchen, W. Germany
(089) 4613-0

FirsTime for Turbo Pascal $74.95

FirsTime for MicroSoft & IBM Pascal $245

FirsTime for C $29

FirsTime is a trademark of Spruce Technology Corporation • MS-DOS is a trademark of Microsoft Corporation
IBM is a trademark of International Business Machines, Inc. • Turbo Pascal is a trademark of Borland International

Dr. Dobb's Journal

Editorial
Editor-in-Chief *Michael Swaine*
Managing Editor *Frank DeRose*
Technical Editor *Alex Ragen*
Editorial Assistant *Sara Noah*
Contributing Editors *Robert Blum, Dave Cortesi, Ray Duncan, Allen Holub*
Copy Editor *Polly Koch*
Typesetter *Jean Aring*

Production
Design/Production Director *Detta Penna*
Art Director *Shelley Rae Doeden*
Production Assistant *Alida Hinton*
Cover Artist *William Cone*

Circulation
Sub. Fulfillment Mgr. *Stephanie Barber*
Subscription Mgr. *Maureen Snee*
Book Marketing Mgr. *Jane Sharninghouse*
Single Copy Sales Mgr. *Kathleen Boyd*

Administration
Finance Manager *Sandra Dunie*
Business Manager *Betty Trickett*
Accounts Payable Supv. *Mayda Lopez-Quintana*
Accounts Payable Asst. *Denise Giannini*
Billing Coordinator *Laura Di Lazzaro*
Administrative Coordinator *Kobi Morgan*

Advertising
Advertising Director
Shawn Horst (415) 424-0600
Advertising Sales
Walter Andrzejewski (617) 567-8361
Lisa Boudreau (415) 424-0600
Beth Dudas (714) 643-9439
Michele Beaty (317) 875-0557
Advertising Systems Manager
Ron Copeland (415) 424-0600
DDJ Classifieds
Tim Ortiz (415) 424-0600
Advertising Administrative Manager
Alice Abrams (415) 424-0600

M&T Publishing, Inc.
Chairman of the Board *Otmar Weber*
Director *C.F. von Quadt*
President *Laird Foshay*

Entire contents copyright © 1985 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0278-6508**

Subscription Rates: $25 per year within the United States, $46 for airmail to Canada, $62 for airmail to other countries. Foreign subscriptions must be pre-paid in U.S. Dollars, drawn on a U.S. Bank. For subscription problems, call: outside of CA 800-321-3333; within CA 619-485-9623 or 566-6947.

Foreign Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 686-1520 TELEX: 620430 (WUI)

People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.

August 1985
Volume 10, Issue 8

CONTENTS

This Month

This month *DDJ* focuses its attention on C. The issue should contain a little something for everybody. One "little something" for all our readers is Jim Hendrix' "Small-C Update." After reading this latest chapter in the story of the compiler first published in the pages of *Dr. Dobb's*, MSDOS C programmers can go straight to the review of MSDOS C compilers, while CP/M C programmers can flip to David Fox' article on optimizing assembly language output from the Software Toolworks' C/80. Unix gurus, on the other hand, will probably want to turn to this month's C Chest, where they can find an MSDOS version of one of their favorite Unix utilities, make. We hope you will find something you like.

Writers' Guidelines

What are the topics on which *DDJ* would like to receive articles? What style of writing is appropriate to *DDJ*? How should I format my text? How should I format listings and how extensively should I comment them? Should I submit hard copy of my manuscript, or a diskette, or both? How much will I get paid for my contribution? The answers to these and other frequently-asked questions can be found in the latest revision of the Writers' Guidelines for *Dr. Dobb's Journal*. To obtain a copy along with the current editorial calendar, drop us a note.

This Month's Referees

Richard Relph, ELXSI
Richard G. Larson, University of Illinois at Chicago
James Hendrix, University of Mississippi

Dr. Dobb's Journal

ARTICLES

C Compilers for MSDOS — 30
by the PicoNet C-SIG and the SVCS
The C-SIG and SVCS have assembled a suite of 30 benchmarks, run them through the most popular MSDOS compilers and interpreters and tabulated the results. **(Reader Ballot No. 191).**

A Peephole Optimizer for Assembly Language Source Code — 56
by David L. Fox
Using regular expressions you can optimize the output of any compiler producing assembly language code. **(Reader Ballot No. 192).**

Small-C Update — 84
by James Hendrix
An update on the compiler first published in the pages of *DDJ*: bug fixes, improvements and MAC, a new Small-C macro assembler. **(Reader Ballot No. 193).**

Asynchronous Protocols — 92
by David W. Carroll
A description of the features, strengths and weaknesses of the most popular asynchronous protocols in use today. **(Reader Ballot No. 194)**

COLUMNS

Dr. Dobb's Clinic — 20
by D. E. Cortesi
DRI Generic Sales; the Z80; nesting MSDOS; quasi random number generators. **(Reader Ballot No. 190).**

C Chest — 96
by Allen Holub
Allen Holub reviews three MSDOS implementations of the Unix utility "make" and offers one of his own. **(Reader Ballot No. 195).**

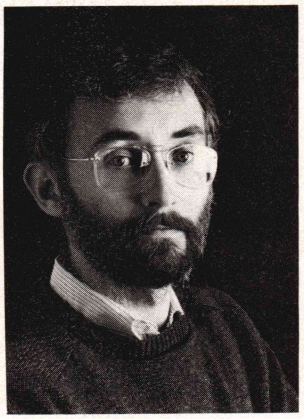
16-Bit Software Toolbox — 116
by Ray Duncan
A bug in TEE; musings on DRI's GEM-Paint and Concurrent PCDOS; square roots; more Mac Feedback. **(Reader Ballot No. 196).**

CP/M Exchange — 124
by Robert Blum
The CP/M Exchange RCP/M; a catalog of free CP/M software; exiting properly to CP/M. **Reader Service No. 197).**

DEPARTMENTS

Editorial 6
Letters 10
DDJ Classified 89
Advertiser Index 128

EDITORIAL



We'd like to congratulate Microsoft, Lifeboat, and PC World for their jointly-sponsored C-85 conference in San Francisco in May. All the *DDJ* editorial staff attended (Allen Holub, our C columnist, and Richard Relph, coordinator of the C compiler review in this issue, were among the speakers), and we found it educational and well-planned.

One session was of particular interest to us. At a discussion of compiler and library evaluation and benchmarking, a speaker chastised computer publications for the inadequacy of their technical reviews. We sat in the audience nodding in agreement.

It's true: today's computer magazines typically do not provide adequate information to permit software developers to choose one compiler or programming tool over another. A relatively conscientious reviewer may (1) run three benchmark tests, usually a floating-point test, the Fibonacci and the Sieve of Eratosthenes; (2) list the product's features; and (3) give a clear picture of his own experiences in using the product. Most do less.

The C-85 conference session pointed out the need for a higher level of criticism, at least in the evaluation of technical products. This is an area in which the vendors and customers are both highly knowledgable. Programmers and system designers read reviews for solid information on which to base decisions affecting their livelihoods. Vendors deserve an accurate appraisal of their products in terms that matter to their customers.

A useful compiler review should, we think, assess completeness, correctness, adherence to standards, and use a widely-available suite of benchmarks applied in a repeatable manner to measure all the major aspects of compile- and run-time performance. Publication of a review should constitute a commitment to continue to evaluate the product, and to publish updates and corrections when appropriate. And no review should be the work of a single individual, no matter how knowledgable or above reproach.

We've tried to provide such a useful review in this month's evaluation of 13 C compilers, done by over a dozen programmers using several dozen benchmarks, with several pairs of eyes reading the manuscript. But we don't claim we brought it off flawlessly, and we'll be following up on this review with updates, corrections (if necessary), and comparable benchmark results for other compilers.

For us, and we think for computer publications generally, this is new ground. we don't believe that any computer magazine has been doing technical reviews the way they should be done, and that includes *Dr. Dobb's Journal*. We hope to change that, and this issue's C compiler review is one step in that direction.

We plan to take some more steps in the direction of useful reviews in the coming months. One desirable feature that this month's C compiler review lacks is a single benchmark test providing a real-world mix of functions; we're looking into some tests that purport to do this. We welcome your feedback and suggestions.

Michael Swaine

Michael Swaine

INTRODUCING
THE MOST RADICAL
ALGORITHM
IN DATABASE HISTORY.

COMMON SENSE.

COMMON SENSE IN DATABASE MANAGEMENT
NUMBER ONE IN A SERIES.

IT'S NOT HOW HARD YOU WORK. IT'S HOW MUCH YOU GET DONE.

There's an odd contradiction built into most database software. Most of the features that make it powerful enough to do the job make it a real struggle to use.

But at Microrim,® we've never thought it makes sense to do things the hard way. That's why we're introducing R:base™ 5000. With a brand new feature that lets you create programs up to ten times faster.

CLIENT DATABASE MENU
1. Add a client master list.
2. Change a client master record.
3. Delete a client master record.
4. Print client report.
5. Exit.

PRODUCTIVITY:
R:base 5000 vs. dBASE III

Using R:base 5000 and dBASE III,™ we built this menu and linked it to its subroutines. The resulting applications were equivalent. The effort required wasn't.

	R:base 5000	dBASE III
Keystrokes	434	6588
Command Lines	47	244
Automated steps	37	0
Time*	9 minutes	2 hours

**Your actual time may vary depending on skill level*

THE APPLICATION EXPRESS.™ WHAT A DIFFERENCE.

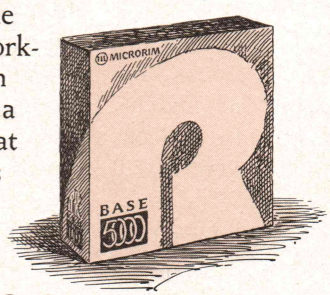
Since the Application Express automatically generates programming code, it can give you the tremendous advantage we've documented in the attached chart. If you've never programmed before, these automated steps can make all the difference between getting the job done and giving up completely. And if you're an experienced programmer, the Application Express can give you ten to one productivity gains. Of course, R:base 5000 also gives you the powerful procedural language and report writer it takes to create highly customized applications.

SEE FOR YOURSELF: 1-800-547-4000.

If you believe in common sense as much as we do, you won't take our word for it; you'll get your hands on a copy and make up your own mind. And that's just what we'd like you to do. For only $9.95 (plus shipping) we'll send you a mini-version of the product that lets you build real-life applications.

Just call 1-800-547-4000 and ask for Dept. 991 . From Oregon, or outside the U.S., call 1-503-684-3000, Dept. 991 . We'll send your copy right out. If you'd like to see R:base 5000 today, head straight for a leading software store or computer dealer. If you own R:base 4000, ask your dealer for a trade-up kit.

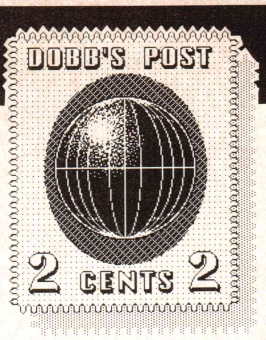
We'll show you how easy it is to get a handle on your workload. When you've got a progam that does things your way.



R:BASE 5000
FROM MICRORIM

IT ALL COMES DOWN TO COMMON SENSE.

dBASE III is a trademark of Ashton-Tate.



Tiny BASIC

Dear *Dr. Dobb's*:

I read with great interest the article "Tiny BASIC for the 68000," which ran in the February 1985 issue of your magazine. I have been a long-time 68000 programmer and have written a full BASIC interpreter for the machine. But, dear Doctor, a 68000 BASIC only about 10% faster than a Z80? I quickly got a copy of the source from the bulletin board and started probing. Mr. Brandly did a nice job of coding, but he missed a few tricks. It is relatively easy to do a substantial speed-up without increasing size significantly or changing any outward functionality.

The main problem in this BASIC implementation is the TSTC subroutine. TSTC compares the current character against a test value and branches to a specified location if they do not match. TSTC gets used a lot in even simple statements; 10 X=1 (crlf) calls it 7 times. I count 98 clocks for TSTC if the characters match, and 102 clocks if they do not match. Inline code to replace TSTC could be made a macro as follows:

```
TSTC  MACRO
      BSR IGNBLK
      CMPI.B #'&1',(A0)
      BNE &2
      ADDQ.L #1,A0
      ENDM
```

which executes in 46 clocks (for match) and 40 clocks (no match). This gives a savings of 52 to 62 clocks per use of TSTC . . . or almost 400 clocks in a simple assignment statement. Any statement involving an expression will see a similar (or greater) improvement.

The second improvement also occurs in the expression evaluator routine EXPR. (This is where most improvements are to be found, since expression evaluation is key to the interpreter's performance.) EXPR uses the EXEC subroutine to check for relational operators (=,<,>,<=,>=). EXEC is a table-searching routine normally used to check for keywords. It includes a lot of overhead to check for shorthanded keywords that can never occur in relationals. Also, the most usual case—no relational—requires searching the six possible relationals first before discovering the default condition. My suggested improvement treats relational operators as a special case, removing them from the EXEC tables and substituting the code in Listing One (page 16) for the EXEC call at the top of EXPR. Note that the default case (no relational) now takes only three compares. This is significant because it is used every time an expression (number, variable, etc.) is encountered.

The third improvement is found in the find-lines routines: (FNDLN, FNDLNP, FNDNXT, and FNDSKP.) These routines find heavy usage in GOTOs, GOSUBs, and IFs. The point here is that the original coding resets register A2 to the end of the program text for each line. This 24 clock overhead per line can be saved by a little re-ordering and a few extra bytes Listing Two, page 16.)

My final improvements do not affect the execution speed of the interpreter, but rather speed up program editing. The block-move subroutines MVUP and MVDOWN can be speeded up quite a bit at the cost of some additional code. I'll illustrate with MVUP. It is currently coded:

```
MVUP   CMP.L A1,A3
       BEQ MVRET
       MOVE.B
         (A1)+,(A2)+
       BRA MVUP
*
MVRET RTS
```

This requires 36 clocks per byte moved. If one calculates the number of bytes to be moved outside the loop, a much tighter inner loop (22 clocks per byte) is possible (see Listing Three, page 17). The set-up overhead is 48 clocks, so it takes only 4 cycles through the inner loop to outweigh it. This version of MVUP can handle a maximum of 65,536 bytes (the original had no restriction), but this should not prove much of a problem in Tiny BASIC programs.

A final note to the Doctor: please keep articles like this coming! There is nothing that improves the "breed" of programmers like quality examples. Thank you.

Sincerely yours,
Robert D. Grappel
28 Buckmaster Drive
Concord, MA 01742

Ellipses

Dear *DDJ*,

Many thanks to Tom Hogan for his excellent derivation of the DV (decision variable) method for plotting ellipses. His explanation of this potentially confusing subject is very clear. This general approach, sometimes called 'displacement comparison' appears elsewhere in the literature, but this is the first time I have seen it applied to ellipses.

I would like to point out a minor error in the derivation, and then show how the full potential of the method can be achieved by one more refinement that eliminates all multiplication within the loops.

The error is in Equation 10. If the x-axis is the major axis, the initial point is at

$$(0, \sqrt{\frac{\beta}{\alpha}}\ R_m), \text{ not } (0, \frac{\beta}{\alpha}\ R_m)$$

Equation 10 is correct if the first term is changed to reflect this. The

The Ultimate Programmer's Editor

WENDIN'S XTC™

VERSION 3.0 STILL ONLY $99.00

SUPER PROGRAMMERS edit in XTC to make software development a snap! Just look at these powerful features:

MULTITASKING

XTC's built-in multitasking lets you run your macros in the foreground or independently in the background while you continue editing. A background process has full access to editor resources, and can be used to translate code from one language to another in REAL TIME, print files in the background, or even scan syntax while you type in code. Best of all, you can use XTC to edit source and documentation in any programming language!

COMPILE IN WINDOWS

All DOS compilers and utilities can be executed from within XTC using a single keystroke. While it runs, XTC captures your compiler's output and redirects it into your text, so you can compare compiler messages with your source code ON THE SAME SCREEN. And using XTC's macro language, Turbo Pascal is literally only a keystroke away. You can use other compilers and utilities inside XTC too — like Lattice "C," Microsoft Pascal, and IBM's Basic, to name a few.

MACRO LANGUAGE

XTC has the most powerful macro language in the editing world. XTC's macros aren't just keystrokes assigned to keys; they're real programs that can be used to automatically edit source code and data files. Like any real programming language, XTC has control structures like IF THEN ELSE, WHILE DO, REPEAT UNTIL, FOR NEXT, DUPLICATE N TIMES, INDEFINITE LOOP, EXIT, and BREAK LOOP. XTC also has INTEGER, BOOLEAN, and STRING variables to hold numbers, conditions, and pieces of text.

WINDOWS & BUFFERS

With XTC you can display up to 8 different files or parts of the same file on the screen at once. XTC's windows are programmable and can even be linked together to share files. XTC also has 20 other buffers that you can use to hold files and blocks of text.

WORDSTAR COMPATIBILITY

If you already know Wordstar commands, then you don't need to learn a new set of commands. If you want to customize XTC, just write macros to emulate the key layout you're used to. XTC can also read Wordstar files, and can even strip off all of the non-standard high bits with a single command.

LARGE FILE EDITING

XTC lets you edit files entirely in memory (using all available memory), or paged to disk, for maximum flexibility. You can choose how XTC buffers text.

XTC outperforms any other programmable editor on all IBM/PC, /XT, and /AT computers (and true compatibles). If you want to feel the power of XTC before you buy it, just ask for our demo disk (only $10) and try it out. When you buy XTC, we'll knock 10 bucks off the price.

To get your copy of XTC now, order it over the phone — we can ship it the same day you call! Or, send in an order, just like this one:

XTC	**$99.00**
Macro Compiler	**FREE**
Shipping, Handling, Insurance	**3.50**
Want it COD? Add this	**1.90**

TOTAL IT UP, AND SEND IT QUICK!

INCREDIBLE EXTRAS!

- UNDO N TIMES
- REMOVE WORDSTAR HIGH BITS
- EDIT GRAPHICS DISPLAYS
- AUTOINDENTING MODE
- TAB EXPANSION/ COMPRESSION MODE
- EXTRA LONG LINES
- MACRO COMPILER
- TELEPHONE SUPPORT
- 150 PAGES OF DOCUMENTATION
- RUNS ON IBM / PC, XT, AND / AT COMPUTERS (AND TRUE COMPATIBLES)

COMPLETE SOURCE CODE

XTC comes with 7,000 lines of source code jam-packed onto two DSDD disks. Includes 13 modules written in Pascal, and 2 assembly libraries you can use to access the PC's screen, intercept software interrupts (like INT 21H functions), allocate and deallocate memory, and load and execute programs. It's all included FREE for your recreation and enjoyment!

INCLUDED FREE! Macro Compiler with full source code in "C." Translates Ada control statements into XTC macros. It even runs in XTC windows!

ORDER HOTLINE
509/235-8088
CREDIT CARDS WELCOME!

WENDIN™

BOX 266
CHENEY, WA 99004

The people who make quality software tools affordable.

Ada® is a registered trademark of the U.S. Department of Defense. Turbo Pascal is a trademark of Borland, Inc. XTC is a trademark of Wendin, Inc.

Circle **no. 112** on reader service card.

For everyone who ever tried doing five things at once

The perfect computer program for someone as busy as you. It lets you keep several other programs working at once.

Do you ever go in so many directions so fast not even a computer can keep up with you?

Well, now an IBM Personal Computer can—thanks to IBM TopView.

TopView is a new kind of software that lets you switch between other programs as quickly as you can change your mind, even run several programs at the same time.

Once you load TopView into your computer, you load the other programs you use most—as many as your computer's memory will permit.

After that, the greatest distance between two programs is just a couple of keystrokes, or (optional) mouse moves.

There's no waiting and a lot less diskette swapping.

But when you're *really* busy is when TopView really shines, letting you do many jobs simultaneously.

For example, you can print a letter, while you search a file, while you analyze a spreadsheet, while your clock/calendar reminds you that your automatic dialer is about to place a call for you.

Little Tramp character licensed by Bubbles Inc., s.a.

...IBM presents TopView.

And you can see everything through on-screen "windows" and control it all with easy-to-use pop-up menus.

You can even make unrelated programs work together; say a "Brand Y" spreadsheet with a "Brand Z" word processor.

But simplest of all is a certain "Brand IBM", namely the IBM Assistant Series—for filing, writing, planning, reporting and graphing.

Many other popular programs also work with TopView, and the number is growing.

Naturally, the more computer memory you have, the more TopView can help you. At least 512K is recommended.

And the price is only $149*.

Beyond that, all you need is to be the kind of person who never does a single thing all day, but who wants to do everything, at once.

To learn more, call an IBM marketing representative, or visit an IBM Product Center or Authorized IBM PC or Software Dealer.

For the store nearest you, and a free brochure, call 800-447-4700. (In Alaska and Hawaii, 800-447-0890.)

IBM®

Personal Computer Software

*IBM Product Center price.

Circle **no. 68** on reader service card.

situation is clarified if one considers a different form of the equation of the ellipse:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

where a and b are the radii along the x the y axes, respectively.

If we let $\alpha = a^2$ and $\beta = b^2$, then Mr. Hogan's Equation 1 is equivalent. It is clear that the aspect ratio is b/a, and Mr. Hogan's use of the aspect ratio is consistent, except for equation 10. If the x-axis is the major axis, then $R_m = a$, and the initial point is at b, which is given by $R_m \times$ aspect ratio, or $R_m \times (b/a)$. (We are assuming that the center of the ellipse is at (0, 0)). This is a fairly obvious error, but it could confuse the unwary (me, for example). The statement in the program which initializes d is correct, so I assume Mr. Hogan found the error and simply overlooked this one instance.

Much more interesting is an examination of the operations that are performed within the loops for computing successive dot locations. An example is the computation of 'd' at the end of the first loop

```
(d += two_beta *
   (3 + (rel_x << 1)) )
```

Note that every term in the computation is a constant except for rel_x, and further, that rel_x is incremented by exactly one every time the loop is executed. So let's rewrite that line for clarity:

```
(d += two_beta*3 +
   two_beta*rel_x*2)
```

or in human rather than computer math

$$(d += 6\beta + 4\beta \text{xrel})$$

Now we will cleverly initialize our added variable d_norm to 2β before entering the loop, so every pass we add 4β to d_norm before adding it, in turn, to the former value of d.

All of the multiplication operations within the loops can be replaced by additions in a similar way, as demonstrated in Listing Four (page 18) (we use x, y in place of rel_x, rel_y to denote the local coordinates relative to the center of the ellipse, and we don't bother to maintain the global coordinates row and col, but the result is the same).

I will be happy to discuss this with anyone who wants to call me at (206) 226-3916, or via IEEE COMPMAIL—user name eric.therkelsen.

Again, many thanks to Mr. Hogan for a derivation that is a significant contribution to the literature.

Sincerely,
Eric E. Therkelsen
ETTS, Incorporated
19224 Southeast 164th
Renton, WA 98058

INTRODUCING
DATALIGHT C

The Datalight C Compiler for MSDOS is a full C with all K&R constructs, including bitfields, plus the version 7 extensions. Other features of the compiler are:

$60

- Produces object files (.obj) so just the MSDOS linker is required.
- Floating point performed with 8087 or automatic software floating.
- Over 100 compact library functions with source.
- Compatible with the Lattice C compiler.
- Runs on IBM-PC, and compatibles, running MSDOS 2.0 or later.
- Complete, easy-to-read users' manual with index.
- Highly optimized code for production quality programs.

Datalight
11557 8th Ave. N.E.
Seattle, Washington 98125
(206) 367-1803

Outside USA add $10 shipping. Washington State residents add 7.9% sales tax. VISA and MasterCard accepted.

IBM-PC, a trademark of IBM; MSDOS, a trademark of Microsoft Corp.; Lattice C, a trademark of Lattice Corp.

Circle no. 29 on reader service card.

ATTENTION
C-PROGRAMMERS

File System
Utility Libraries

All products are written entirely in K& RC. Source code included, No Royalties, Powerful & Portable.

BTree Library 75.00

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

ISAM Driver 40.00

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

Make 59.00

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

For more information call or write:

softfocus

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903
(416) 844-2610

Credit cards accepted.

Dealer inquiries invited.

Circle no. 126 on reader service card.

Tools That Make Your Job Easier

For PCDOS/MSDOS (2.0 and above/128K) • IBM PC/Compatibles, PC Jr., Tandy 1000/1200/2000, & others
For CPM80 2.2/3.0 (Z80 required/64K) • 8" SSSD, Kaypro 2/4, Osborne I SD/DD, Apple II, & others

MIX EDITOR

Programmable, Full/Split Screen Text Processor

Introductory Offer **29.95**

Great For All Languages

A general purpose text processor, the MIX Editor is packed with features that make it useful with any language. It has auto indent for structured languages like Pascal or C. It has automatic line numbering for BASIC (255 character lines). It even has fill and justify for English.

Split Screen

You can split the screen horizontally or vertically and edit two files simultaneously.

Custom Key Layouts

Commands are mapped to keys just like WordStar. If you don't like the WordStar layout, it's easy to change it. Any key can be mapped to any command. You can also define a key to generate a string of characters, great for entering keywords.

Macro Commands

The MIX Editor allows a sequence of commands to be executed with a single keystroke. You can define a complete editing operation and perform it at the touch of a key.

Custom Setup Files

Custom keyboard layouts and macro commands can be saved in setup files. You can create a different setup file for each language you use.

MSDOS Features

Execute any DOS command or run another program from inside the editor. You can even enter DOS and then return to the editor by typing exit.

MIX C COMPILER

Full K&R Standard C Language
Unix Compatible Function Library

Introductory Offer **39.95**

Complete & Standard

MIX C is a complete and standard implementation of C as defined by Kernighan and Ritchie. Coupled with a Unix compatible function library, it greatly enhances your ability to write portable programs.

The Best C Manual

MIX C is complemented by a 400 page manual that includes a tutorial. It explains all the various features of the C language. You may find it more helpful than many of the books written about C.

Fast Development

MIX C includes a fast single pass compiler and an equally fast linker. Both are executed with a simple one line command. Together they make program development a quick and easy process.

Fast Execution

The programs developed with MIX C are fast. For example, the often quoted prime number benchmark executes in a very respectable 17 seconds on a standard IBM PC.

Standard Functions

In addition to the functions described by K&R, MIX C includes the more exotic functions like *setjmp* and *longjmp*. Source code is also included.

Special Functions

MIX C provides access to your machine's specific features through BDOS and BIOS functions. The CHAIN function lets you chain from one program to another. The MSDOS version even has one function that executes any DOS command string while another executes programs and returns.

Language Features

- Data Types: char, short, int, unsigned, long, float, double (MSDOS version performs BCD arithmetic on float and double-no roundoff errors)
- Data Classes: auto, static, extern, register
- Struct, Union, Bit Fields (struct assignment supported)
- Typedef, Initialization
- All operators and macro commands are supported

30 DAY MONEY BACK GUARANTEE

Orders Only: **Call Toll Free 1-800-523-9520,** (Texas only **1-800-622-4070**)

MIX Editor ___ $29.95 + shipping ($5 USA/$10 Foreign) Texas residents add 6% sales tax
MIX C ___ $39.95 + shipping ($5 USA/$25 Foreign) Texas residents add 6% sales tax
Visa ___ MasterCard ___ Card # _____ Exp. Date _____
COD ___ Check ___ Money Order ___ Disk Format _____
Computer _____ Operating System: MSDOS ___ PCDOS ___ CPM80 ___
Name _____
Street _____
City/State/Zip _____
Country _____
Phone _____

MIX software
2116 E. Arapaho
Suite 363
Richardson, Tx 75081

Dealer Inquiries Welcome
Call (214) 783-6001

MSDOS is a trademark of Microsoft PCDOS is a trademark of IBM CPM80 is a trademark of Digital Research WordStar is a trademark of MicroPro

D

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support.
  Full range transcendentals
  (tan, sin, cos, arctan,
  logs and exponentials)
  Data type conversion and
  I/O formatting.
- High level interrupt support.
  Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS/FORTH

- Fully Optimized & Tested for:
  IBM-PC IBM-XT IBM-JR
  COMPAQ EAGLE-PC-2
  TANDY 2000 CORONA
  LEADING EDGE
  (Identical version runs on almost all MSDOS compatibles!)
- Graphics & Text
  (including windowed scrolling)
- Music - foreground and background
  includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler
  (interactive, easy to use & learn)
- Compare
  BYTE Sieve Benchmark jan 83
  HS/FORTH 47 sec BASIC 2000 sec
  w/AUTO-OPT 9 sec Assembler 5 sec
  other Forths (mostly 64k) 70-140 sec

FASTEST FORTH SYSTEM AVAILABLE.

TWICE AS FAST AS OTHER FULL MEGABYTE FORTHS!

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: $250.

VISA Visa Mastercard MasterCard

Add $10. shipping and handling

HARVARD SOFTWORKS

PO Box 69
Springboro, Ohio 45066
(513) 748-0390

Circle **no. 44** on reader service card.

Prolog

Sirs:

I enjoyed your March, 1985, issue on Prolog, but thought I should write to warn your other readers of possible support problems with Programming Logic Systems, Inc., the U.S. marketers of MicroPROLOG. Over the past 5 months I have attempted, with no success whatsoever, to obtain the simple information whether there have been any changes or upgrades to the software in the year or so since I purchased it. Neither the U.S. or British offices will respond to repeated mail inquires (the British office simply sent a brochure, in no way answering my very simple question). If these people can't respond to a question as simple as that, it does not bode well for their willingness to respond to substantive inquiries, and suggests to me that one of the other fine implementations of the language would be a better choice.

Sincerely,
George Carey
2048 Winding Creek Ln.
Marietta, GA 30064

DDJ

Letters (Text begins on page 10)

Listing One

```
        BSR IGNBLK              skip blanks
        MOVE.B (A0)+,D2         get first char.
        CMPI.B #'=',D2          equals?
        BEQ XP15                yes, process it
*
        CMPI.B #'>',D2          >, or >= ?
        BNE XXP1                no
*
        CMPI.B #'=',(A0)        >=?
        BNE XP13                no, process >
*
        ADDQ.L #1,A0            skip past =
        BRA XP11                process >=
*
XXP1    CMPI.B #'<',D2          <, <=, or <>?
        BEQ XXP2                yes
*
        SUBQ.L #1,A0            no, default case
        BRA XP17                handle default
*
XXP2    CMPI.B #'=',(A0)        <=?
        BNE XXP3                no
*
        ADDQ.L #1,A0            yes, skip past =
        BRA XP14                process <=
*
XXP3    CMPI.B #'>',(A0)        <>?
        BNE XP16                no, process <
*
        ADDQ.L #1,A0            skip past >
        BRA XP12                process <>
```

End Listing One

Listing Two

```
FNDLN   CMPI.L #$FFFF,D1
        BCC QHOW
        MOVE.L TXTBGN,A1
        MOVE.L TXTUNF,A2        note change
        SUBQ.L #1,A2
*
FNDLNP  CMPA.L A1,A2
        BCS FNDRET
*
        MOVE.B (A1),D2
        LSL.L #8,D2             note change
        MOVE.B 1(A1),D2
        CMP D1,D2
        BCC FNDRET
```

```
*
F1      ADDQ.L #2,A1            note change
F2      CMPI.B #CR,(A1)+
        BNE F2
*
        BRA FNDLNP
*
FNDNXT  MOVE.L TXTUNF,A2        note change
        SUBQ.L #1,A2
        BRA F1
*
FNDSKP  MOVE.L TXTUNF,A2        note change
        SUBQ.L #1,A2
        BRA F2
```

End Listing Two

Listing Three

```
MVUP    MOVE.L D0,-(SP)         preserve D0
        MOVE.L A3,D0
        SUB.L A1,D0             compute no. of bytes
        BLE MVRET               no bytes?
*
        SUBQ.L #1,D0            adjust for DBRA
*
MV2     MOVE.B (A1)+,(A2)+
        DBRA D0,MV2
*
MVRET   MOVE.L (SP)+,D0         restore D0
        RTS
```

End Listing Three

(Listing Four begins on next page)

pcTEX™ is here!

Complete TEX82 Typesetting for your PC/XT or AT

- Real, state-of-the-art typesetting capable of handling all mathematical and scientific material.
- Produce work of this quality on your Epson printer.

$$\sum_{p \text{ prime}} f(p) = \int_{t>1} f(t)\, d\pi(t). \qquad \{\overbrace{a,\ldots,a}^{k\ a\text{'s}},\overbrace{b,\ldots,b}^{l\ b\text{'s}}\}.$$
$$k+1 \text{ elements}$$

- Outperform professional typesetters with work of this quality from a QMS Lasergrafix.

$$G(z) = e^{\ln G(Z)} = \exp\left(\sum_{k\geq 1} \frac{S_k z^k}{k}\right) = \prod_{k\geq 1} e^{S_k z^k/k}$$

- PCTEX includes INITEX, LaTEX macro package and manual, PCTEX macro package and manual.
- PCTEX is a full implementation of Donald Knuth's TEX and produces standard .DVI files.
- PCTEX: only $279. PCDOT dot-matrix printer driver (includes over 200 fonts): $100. (For IBM Graphics printer, Epson RX FX printers, Toshiba 134x 135x.) QMS Lasergrafix driver: $200.

Include $3. shipping for each order. Calif. residents add sales tax. MasterCard, Visa accepted.
Requires DOS 2.0 or better, 512K RAM, 10M hard disk.

PERSONAL TEX INC
20 Sunnyside, Suite H, Mill Valley, CA 94941.
(415) 388-8853. Telex 275611.

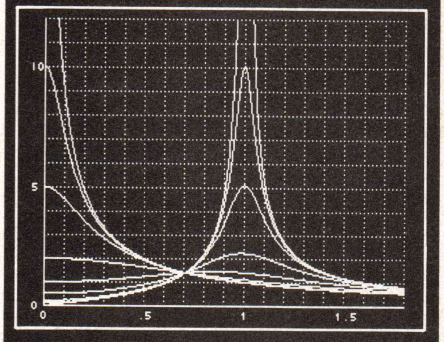
TEX: American Mathematical Society. PCTEX: Personal TEX, Inc., IBM-PC: IBM Corp., QMS: QMS, Inc.

Circle no. 76 on reader service card.

ISYS FORTH

for the Apple®] [

Fixed point speed can rival that of floating point hardware. But the details have been a well kept secret—until now. The following graphs were generated by fixed point examples from the ISYS FORTH manual.



Parallel Resonance with Damping
BASIC 213 sec ISYS FORTH 27 sec

Hydrogen 3p Orbital Cross-section
BASIC 492 sec ISYS FORTH 39 sec

- **Fast** native code compilation. Sieve benchmark: 33 sec
- **Floating Point**—single precision with trancendentals
- **Graphics**—turtle & cartesian with 70-column character set
- **Double Precision** including D*/
- **DOS 3.3 Files** read & written
- **FORTH-83** with standard blocks
- **Full-Screen Editor**
- **Formatter** for word processing
- **Macro Assembler**
- **Price: $99,** no extra charges

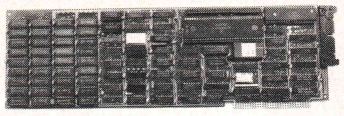
ILLYES SYSTEMS
PO Box 2516, Sta A
Champaign, IL 61820

Technical Information:
217/359-6039, mornings

For any Apple] [model, 48K or larger. Apple is a registered trademark of Apple Computer.

Circle no. 46 on reader service card.

68000 CO-PROCESSING For IBM PC, PC/XT and COMPATIBLE SYSTEMS



Now you can add the MOTOROLA 68000 16/32 Bit Processor to your PC via use of the Pro 68 Advanced Technology Co-Processor. Enjoy all of the performance benefits of the 68000 processor without sacrificing your current PC system. Consider these impressive standard features of Pro 68:

- High Speed MOTOROLA 68000 micro processor
- 10Mhz no wait state design (3 times faster than the IBM PC/AT)
- True 16/32 bit technology
- For use on IBM PC, PC/XT or compatible systems
- On board 16 bit parity checked memory, 256K to 1024K
- Two serial I/O ports for multi user interface
- Provisions for the high speed NS32081 math processor
- High speed proprietary dual port host bus interface
- Parallel or array processing via multi processor architecture
- MS/PC DOS RAM disk driver program
- Choice of two popular integrated 16/32 bit operating systems:
  - — CPM68K from Digital Research Inc.
    - — Full suite of development tools
    - — "C" compiler with floats and UNIX I/O library
    - — Many third party compatible languages and applications
  - — OS9/6800 from MICROWARE Corporation
    - — UNIX look alike with multi user/multi tasking, shell, hierarchical disk directory, record and file lock, pipes and filters
    - — Full suite of development tools
    - — UNIX V compatible "C" compiler
    - — Optional languages include BASIC, ISO PASCAL, FORTRAN 77.

Pricing from $1195 includes Pro68 with 256K, OS, and MS/PC DOS RAM disk driver. HSC also manufactures and markets a full line of co-processors and RAM disks for use on Z80 based systems.

DISTRIBUTORS:

Australia-Computer Transition Systems . . . 03-537-2768
Great Britain-System Science . . . 01-248-;062
West Germany-DSC International . . . 089-723-1125
Canada Remote Systems . . . 416-239-2835

Dealer, Distributor and OEM inquiries invited.

HSC

Hallock Systems Co., Inc.
267 North Main Street
Herkimer, NY 13350
(315) 866-7125

Circle **no. 26** on reader service card.

Letters (Listing continued, text begins on page 10)

Listing Four

```
ellipse_write ( xc, yc, a, aspect )
    int xc, yc, a;
                /* xc, yc are coords of center; a is radius along x-axis */
                /* we chose to input a rather than Rm */

    float aspect;
    {
    long alpha, beta;
    long four_alpha, four_beta, alpha_y, beta_x, d, d_exc, d_norm;
    int b, Rm;     /* b is radius along y-axis */
        /* since a is not necessarily the major radius, we will compute Rm */

    int  x, y;          /*    local coords relative to center of ellipse    */
                                .
                                .

            [ initialization similar to Mr. Hogan's ]

                                .
                                .

    alpha_y = alpha * y;     /* new vbls to avoid multiplication in loop...*/
    beta_x = 0L;
    d_exc = - four_alpha * y;
    d_norm = beta << 1;

    d = 2*alpha*(y*(y-1)) + alpha + 2 * (beta - beta*Rm - alpha*beta);
            /* This is Mr. Hogan's original statement - it is correct
               even though equation 10 is not.    */

    while ( alpha_y > beta_x )
        {
        put_dots ( xc, yc, x, y );
        if ( d >= 0 )
            {     /*    exception condition - decrement y occasionally    */
            d += ( d_exc += four_alpha );
            alpha_y -= alpha;
            y--;
            }
        d += ( d_norm += four_beta ) ;     /*    always increment x    */
        beta_x += beta;
        x++;
        }

    /*    slope is now < -1, so proceed with rest of ellipse    */
                                .
                                .

                    [ 'nuff said? ]
                                .
                                .
```

End Listings

No source code for your REL files?

REL/MAC

converts a REL file in the Microsoft™ M80 format to an 8080 or ZILOG™ Z80 source code MAC file with insertion of all public and external symbols.

- **REL/MAC** makes MAC source files
- **REL/MOD** lists library modules
- **REL/VUE** displays the bit stream
- **REL/PAK** includes all of the above
- 8080 REL/MAC **demo disk** $10.00

REL/PAK for 8080 only $99.95
REL/PAK for Z80 & 8080 ..... $134.95
on 8"SSSD disk for CP/M™ 2.2

Send check, VISA, MC or C.O.D. to

MICROSMITH COMPUTER TECHNOLOGY
P.O. BOX 1473 ELKHART, IN 46515
1-800-622-4070
(Illinois only 1-800-942-7317)

Circle **no. 41** on reader service card.

Disk Sale

dysan **Dysan** CORPORATION

TYPE	BOX OF 10
5"-SS/DD-48 TPI	19.50
5"-DS/DD-48 TPI	25.50
5"-SS/DD-96 TPI	29.50
5"-DS/DD-96 TPI	37.50
5"-DS/DD-IBM/AT	52.95
8"-SS/SD-48 TPI	23.95
8"-SS/DD-48 TPI	25.50
8"-DS/DD-48 TPI	29.95
3.5"-SS/DS	32.95

Available Soft or Hard Sector
For Plastic Case Add 1.25/Box
Plus Tax & Shipping
- Cash, Visa, Mastercard, COD -

Integral Systems Corp.
2900-H Longmire Drive
College Station, TX 77840
(409) 764-8017

Circle **no. 23** on reader service card.

Another in a series of productivity notes on MS-DOS™ software from UniPress.

TEXT EDITING

UNIPRESS EMACS™

Subject: Multi-window full screen editor.

Multiple windows allow several files (or portions of the same file) to be edited simultaneously. Programmable through macros and the built-in compiled MLISP™ extension language.

Features:

- *Famed Gosling Version.*
- *Extensible through the built-in MLISP programming language and macros.*
- *Dozens of source code MLISP functions; including C, Pascal and MLISP syntax checking.*
- *EMACS runs on TI-PC™, IBM-PC AT™ DEC Rainbow™ or any other MS-DOS machine. Requires at least 384k.*
- *Run Lattice® C or PsMake™ in the background and EMACS will point to any errors for ease of debugging. PsMake is a UNIX™-style "make" utility to automate the process of building complex programs.*
- *Optional Carousel Tools: UNIX-like facilities including cat, cp, cd, logout, ls, mv, pwd, rm, set, sh and more.*

Price:

EMACS	*$475*
One month trial	*75*
Available for UNIX and VMS.	
PsMake	*179*
Carousel Tools	*149*
Full System	*1,299*

Includes EMACS (object), PsMake, Lattice C, PHACT™ ISAM and Carousel Tools.

COMPILER FOR THE 8086™ FAMILY

LATTICE® C COMPILER

Subject: Compiler for MS-DOS.

Lattice C Compiler is regarded as the finest compiler for MS-DOS and is running on thousands of 8086 systems.

Features:

- *Runs on the IBM-PC™ under MS-DOS 1.0, 2.0 or 3.0*
- *Produces highly optimized code.*
- *Small, medium, compact and large address models available.*
- *Standard C library.*
- *PLINK—optional full function linkage editor including overlay and support.*

Price:

Lattice C Compiler	*$425*
PLINK	*425*

ISAM FILE SYSTEM

PHACT™

Subject: Powerful Keyed File Access for MS-DOS.

PHACT ISAM is a keyed B+ tree file manager providing easy access to and manipulation of records in a database.

Features:

- *Supports fixed and variable length records (1-9999 bytes).*
- *Up to 9 alternate indices are supported.*
- *Record locking allows each record in the database to allow multiple simultaneous updates.*
- *Records can be accessed on full or partial key.*
- *Includes full Lattice linkable library and high-level functions.*

Price:

PHACT ISAM	*$250*

Source Code available, call for terms.

For more information on these and other UNIX software products, call or write: UniPress Software, Inc., 2025 Lincoln Hwy., Edison, NJ 08817. Telephone: (201) 985-8000. Order Desk: (800) 222-0550 (Outside NJ). Telex: 709418. Japanese Distributor: Softec 0480 (85) 6565. European Distributor: Modulator SA (031) 59 22 22.

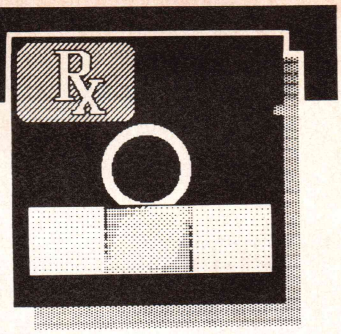
OEM terms available. Mastercard/Visa accepted.

Trademarks of: Lattice, Lattice, Inc., UniPress EMACS and MLISP, UniPress Software, Inc., MS-DOS, Microsoft, UNIX, AT&T Bell Laboratories, Carousel Tools, Carousel MicroTools, Inc., PHACT, PHACT Associates, 8086, Intel, TI-PC, Texas Instruments, IBM PC/AT, International Business Machines, DEC Rainbow/VMS, Digital Equipment Corp

UniPress Software

Your Leading Source for UNIX Software.

DR. DOBB'S CLINIC



by D. E. Cortesi

DRI Generic Sales Again

Barry Watzman writes to contradict us regarding Digital Research's policy on selling generic operating systems. He claims that DRI is "not only not selling generic systems any longer, they are no longer licensing value-added vendors either. The new rule is, if you're not the manufacturer of the computer on which the implementation will run, forget it."

By way of proof, he sent along a photostat of a letter to him from Alice Clark, Contract Administrator at DRI. It includes the sentence: "We are not currently licensing companies to modify and redistribute our software unaccompanied by hardware or an application program."

The dispute seems to be over how much and what kind of value a "value-added" vendor must add to be acceptable. Watzman wants to make a business of configuring DRI systems to Zenith hardware (he does it very well from all we hear). DRI's position, he says, prevents it.

What Z800?

Roger Smith was turning over back issues of *DDJ* when his eye was caught by the confident words of L. Barker in this column for November 1981. "When the Z800 comes out next year," Barker averred, "with the Z80 instruction set and multiply and divide, this [worrying about the speed of arithmetic] will look like a waste of time."

That pin-compatible Z80 superchip never appeared, so far as we know. "Does anybody know what happened?" Smith wonders, and so do we.

Nesting MSDOS

Remember how we finally got CP/M batch command processing pretty well fixed up? If you don't, you might like to refer to these old Clinics:

June 1982 for PAUSE and BEEP commands
August 1982 for null-line, lowercase, and control character fixes
September 1982 for the QUITIF program
October 1982 for nested submits
January 1984 for a PAUSE that works in CP/M Plus

You might think that batch execution in MSDOS Version 2 wouldn't need our help, but you'd be wrong: MSDOS won't nest batch files. When you invoke a batch file in a batch file, the called file replaces its caller instead of being its subroutine.

We just discovered how to do it. You nest batch files by nesting command shells. Command.com, the shell, can be named as a command. If given a /c switch, it executes the single command that follows on the command line and ends. If that command is a batch file, it will be executed in full, and control will return to the calling batch file. Type in our demo file, recurse.bat (Listing One, page 24), and try it out with a line like

```
recurse 0 1 2 3 4 5
```

What's nice is that each level of command.com has its own environment. The nested batch file can do all the sets and paths it likes; on exit, the prior environment is restored. Add a chkdsk command to the end of the file to see how much space each nesting level takes up.

Random Error

In February, we published a quasi-random number generator (QRNG) credited to persons named Talley and Metropolis (about whom we know nothing more). Ever since then, we've been backing away from it, but we have the subject more or less under control now, mostly thanks to David Ross and Michael Barr.

We showed the QRNG in C. It had two main parts. The first, rand0( ), purported to produce quasi-random numbers uniformly distributed between zero and MAX. Take that at face value; we'll come back to it.

The second part, randi(*u*), used the first to produce what purported to be quasi-random numbers uniformly distributed between 1 and *u*. It did not. Here's the essence of how we coded it:

```
randi(u)
unsigned u;
{
    return(1 + rand0( ) % u);
}
```

The percent sign is the C sign for modulus. Can you see what's wrong with this?

Michael Barr and Peter K. Pearson did. Barr put it best: "Imagine you adopted the following method to find random digits from 0 to 9: throw a dart at the face of a clock and take the low-order digit of the number nearest to where the dart hits. It is evident that you will get the digits 1 and 2 one time in six and the other digits once in twelve. Thus a random event is translated into one that is still random, but not uniformly distributed. [Your procedure randi(*u*)] does that. It shows up most clearly when *u* is a sizable fraction, say 2/3, of MAX. In that case, you will get approximately twice as many values between 0 and *u* /2 as between *u* /2 and *u*."

We devised randi( ) with the vague mental image of how the mod function folds the real number line, like a carpenter's rule, into a heap of *u*-sized segments. And so it does, but

the short last piece improperly enriches the low end of the range. What randi( ) really wants to do is to rescale the range 1 . . . MAX to fit in the interval 1 . . . *u*. Rescaling calls for, not mod, but division, like this:

```
randi(u)
unsigned u;
{
  return(1+ rand0( )*u/MAX);
}
```

But there are problems implementing that. The main charm of these routines was that they worked with 16-bit integers. The expression *u* /MAX is a fraction; if we do rand0( )**u* first, its result will exceed 16 bits.

Now, rand0( ) makes use of two constants: MAX0 and MAX. MAX0 is usually 2^w, while MAX = MAX0–1. In our original version, *w* was 15, but clearly it should be the word size of the computer, usually 16. Supposing that to be the case, the expression

```
rand0( )*u/MAX0
```

can be implemented cheaply enough by doing a 16-bit unsigned multiply and taking the most significant 16 bits of the 32-bit result. Picture it as using multiplication by *u* to squeeze out some of the high-order bits of rand0( ).

The right scaling factor is *u*/MAX not *u*/MAX0. But David Ross says that "to adjust *xu*/MAX0 to make it equal *xu*/MAX, just add one whenever *xu*/(MAX0**MAX) is greater than 0.5, *i.e.* when 2*xu* exceeds MAX0**MAX."

All these binary manipulations make it impractical to present the revised rand0( ) and randi( ) in any language but assembly language. Versions for the 8086 and Z80 follow. If we had versions for the 6xxxx machines, we'd print 'em and will if you send us some.

Considering QRNGs

The strongest impression we've gained from this exercise in QRNGs is of our own ignorance. A *lot* is known about the design of QRNGs, and most of it is strongly mathematical. The best source we know is Donald Knuth's *Seminumerical Algorithms*, pages 1–178. There oughta be a law that nobody can write about QRNGs until they pass a midterm on this material—in which case, this column would be on another topic even though we spent several hours recently poring over that chapter.

We learned quite a bit about linear congruential QRNGs, but the Talley-Metropolis routine is not one of those. Mathematically, it can be stated as:

$$X_n = ((X_{n-1} + X_{n-2}) \bmod \text{MAX0}) * 2) \bmod \text{MAX}$$

which is not identical to any of the QRNG formulae Knuth treats (although it bears an uncomfortable resemblance to the Fibonacci sequence, whose output Knuth says "is definitely *not* random").

We also learned that you should *never* base an application on a QRNG that hasn't been formally justified and thoroughly tested. Therefore you shouldn't rely on this one. We publish it because readers might enjoy working *on* it, as opposed to *with* it.

We also learned a lot about testing QRNGs; Knuth is particularly good on that subject. In particular, we

Mystic Pascal

Fastest Compiler on Earth—$39.95!

Mystic Pascal compiles at 100,000 to over 1,000,000 lines per minute! How? It takes a short cut called *incremental compilation*. Compared to earlier Pascals, the effective speed is astronomical. Give the Compile command for a 1000 line program, and you probably can't lift your finger from the keyboard before the compiler flashes—DONE!

640K of storage not 64K. Are you fed up with being forced to shoehorn your programs into 64K? You won't need mystical powers to run your program in the full 640K that we allow—code, data and stack.

Interactive Pascal. You can enter Pascal statements directly and see the results instantly. It works like a Basic interpreter but it's a true compiler!

Optimized 8086 Code. Mystic Pascal produces true 8086 object code. The TWO code optimizers run in the background so they don't slow you down. Thanks to another breakthrough, our software floating point arithmetic runs faster than Intel says is possible on the 8088/8086 chips.

Real Multi-Tasking Pascal. Advanced programmers may write truly concurrent Pascal programs by simply starting Pascal procedures. Up to 100 concurrent procedures can communicate by passing messages through queues.

ISO Standard Pascal. Educators in particular need a truly Standard Pascal for their students. And learning is made easier by the Help Windows which describe Pascal.

Mystic Canyon Software
P.O. Box 1010
Pecos, New Mexico 87552

Place your order by phone today—(505) 988-4214 or mail the coupon.
Requires an IBM Personal Computer or true compatible with 256K. Not copy protected.

Rush me the Mystic Pascal System with diskette and manual!

Name _____

Address _____

City _____ State _____ Zip _____

☐ Check/Money Order ☐ Visa ☐ Mastercard

Price is $39.95 plus $4 shipping. CODs and Purchase Orders are NOT accepted. Outside US & Canada shipping charge is $20. Payment must be in US funds on a US bank.

Card _____ Exp. _____

Signature _____

Circle no. 79 on reader service card.

know that you can spend about as much time as you like coding, running, and analyzing tests. The time we could spend on this project was severely limited. But here are what we and our correspondents have learned.

Period Length

A major point of interest is the length of the sequence of numbers the QRNG will produce. Michael Barr's approach to this was interesting and could be the basis of more extensive theoretical and practical work. "You can think of it as tracing an orbit among the sets of pairs of numbers (a,b) with a and b in the range 0 . . . MAX. And you can ask the question, what is the length and nature of that orbit?

"Clearly, (0,0) is an orbit of its own. In particular, the generator can never produce two zeroes in a row [*really? DEC*]. This is, by itself, a failure of randomness, perhaps not a serious one, but nonetheless a failure.

"When I tried it with MAX0 = 8, I discovered that the other 63 pairs were all in one orbit! This was surprising, but when I tried out other values of MAX0 (all even) I found that this is a fluke. For no other small value of MAX0 do all the nonzero pairs make up a single orbit. For example, for MAX0 = 6, there is an orbit of length 14, one of 16, and two of length 1—(0,0) as always and (4,4). As a matter of fact, if MAX0 is divisible by 6, there is always that second fixed point."

We'd been operating under the assumption that a generator that produced 16-bit numbers would have a maximum period of 65,536, but that proved not to be the case. We hacked together a version of what Knuth calls the collision test. This program cleared 64K bits (8K bytes) then called Rand0 256 times and set the selected bits to one. Then it began getting and counting Rand0 values and testing the bits they indexed. It counted successive hits of 1-bits.

We had presumed that, if the QRNG were working right, it would generate 64K numbers and then start repeating, hitting all 256 1-bits in a row; or it might hit them all again sooner, indicating a short sequence from the original seeds; or it might get into a loop hitting 0-bits and never return to the original 256 numbers.

Nothing of the sort happened. It behaved instead like a well-distributed generator with a much, much longer period. It made single hits at intervals of roughly 256 probes; it made double hits at intervals of roughly 64K; and when we stopped after 1.5 million probes, it had yet to hit three 1-bits in a row (intuitively, that should happen around 24 million), nor had it looped.

This is an intriguing result. A QRNG with such a long period that codes so tightly in a 16-bit machine is a rare bird. We only wish that somebody would analyze its period from a theoretical standpoint. It would be almost as nice if somebody would carry out a longer collision test.

A question related to period length is: What is a good initial seed? The QRNG is initialized with two numbers; what should they be for a maximal period? It's all very well to stick in two 16-bit prime numbers; if we had any theoretical understanding, we might be able to specify rules for the selection of good seeds.

Distribution

Of equal concern is the distribution. A few short tests have been made. Everett Carter generated 1000 numbers using randi(100) with the flawed randi( ) function. He subjected them to a Chi-squared test with 99 degrees of freedom; the result was a probability of 0.82 that the sample was drawn

CP/M-80 C Programmers . . .

Save time

. . . with the BDS C Compiler. Compile, link and execute *faster* than you ever thought possible!

If you're a C language programmer whose patience is wearing thin, who wants to spend your valuable time *programming* instead of twiddling your thumbs waiting for slow compilers, who just wants to work *fast,* then it's time you programmed with the BDS C Compiler.

BDS C is designed for CP/M-80 and provides users with quick, clean software development with emphasis on systems programming.

BDS C features include:

- Ultra-fast compilation, linkage and execution that produce directly executable 8080/Z80 CP/M command files.
- A comprehensive debugger that traces program execution and interactively displays both local and external variables by name and proper type.
- Dynamic overlays that allow for run-time segmentation of programs too large to fit into memory.
- A 120-function library written in both C and assembly language with full source code.

Plus . . .

- A thorough, easy-to-read, 181-page user's manual complete with tutorials, hints, error messages and an easy-to-use index — it's the perfect manual for the beginner *and* the seasoned professional.
- An attractive selection of sample programs, including MODEM-compatible telecommunications, CP/M system utilities, games and more.
- A nationwide BDS C User's Group ($10 membership fee — application included with package) that offers a newsletter, BDS C updates and access to public domain C utilities.

Reviewers everywhere have praised BDS C for its elegant operation and optimal use of CP/M resources. Above all, BDS C has been hailed for it's remarkable *speed.*

BYTE Magazine placed BDS C ahead of all other 8080/Z80 C compilers tested for fastest object-code execution with all available speed-up options in use. In addition, BDS C's speed of compilation was almost *twice* as fast as its closet competitor (benchmark for this test was the Sieve of Eratosthenes).

"I recommend both the language and the implementation by BDS very highly."
Tim Pugh, Jr.
in *Infoworld*

"Performance: *Excellent.*
Documentation: *Excellent.*
Ease of Use: *Excellent.*"
InfoWorld
Software Report Card

". . . a superior buy . . ."
Van Court Hare
in *Lifelines/The Software Magazine*

Don't waste another minute on a slow language processor. Order your BDS C Compiler today!

Complete Package (two 8" SSSD disks, 181-page manual): **$150**
Free shipping on prepaid orders inside USA.
VISA/MC, COD's, rush orders accepted. Call for information on other disk formats.

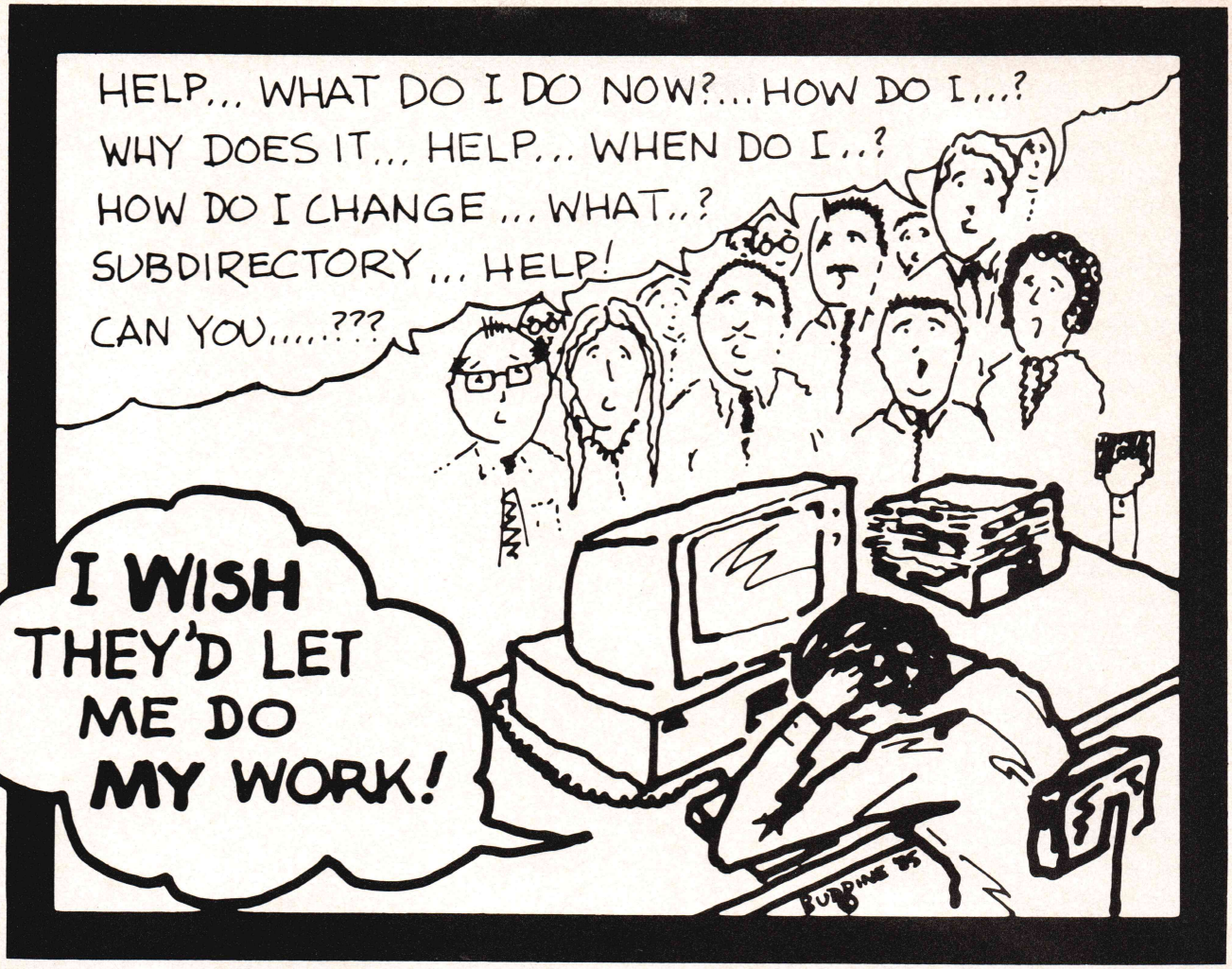
BDS C is designed for use with CP/M-80 operating systems, version 2.2. or higher. It is not currently available for CP/M-86 or MS-DOS.

BDSoftware

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

Circle no. 12 on reader service card.

TIME IS MONEY



Your Wish Is Our MenuCommand™

Here is the perfect productivity tool for use by unsophisticated end users - or for integration into your own application system.

MenuCommand is a menu display editing system for PC/MS-DOS.

It is extremely fast and more powerful in features than any other menu system available. It supports unlimited menus and up to 24 selections per menu. Any menu selection can be specified to run a .COM or .EXE program (with automatic or screen-prompted parameters), any batch file, or any DOS command string up to 60 characters (with automatic or screen-prompted parameters). MenuCommand currently supports both cross-device and cross/sub-directory file access. Includes password protection of menus, DOS access, Menu Editor access, and menu selections (10 levels), and supports custom-color menu displays.

$49.95

Money-Back Guarantee
One Hour Free
Support Included

CSS

Command Software Systems, Inc.
5308 Derry Avenue, Suite K
Agoura Hills, CA 91301
(818) 707-7100

Circle **no. 38** on reader service card.

from a uniform distribution. Says Carter, "if I needed a uniform distribution, I'd try for better than 0.82."

Barr notes that, "provided MAX0 is even, it is possible to run the generator backwards. That is, if you think of it as producing a sequence X_1, X_2, ... X_n, you can calculate X_{n-2} from X_n and X_{n-1}."

Peter Pearson made the remarkable assertion that "sets of numbers drawn three at a time from rand0 are strongly interdependent. Specifically, if p, q, and r are the results of three successive calls on rand0, the quantity $2(p+q)-r$ can have one of only four values: 0, 32767, 65536, and 98303."

But on the other hand, Carter said, "On the correlation test, this generator does very well over short spans (I never tried long spans). This test measures the independence of the numbers by calculating the correlation between one number and a number later in the series. In the tests that I made with a lag distance of up to 20 intervals, the correlations are below 0.1 and most are well under 0.05. While one might hope for more independence, these values are very good relatively."

Clearly many more tests of this QRNG should be done. Knuth recommends the runs test and the collision test as particularly powerful and describes algorithms for them and several others.

The Mitchell-Moore QRNG

There is another simple QRNG, one that has guaranteed good characteristics and a ridiculously long period. Its only drawback is that it needs 55 words of auxiliary space and a second QRNG to seed it. Knuth credits the method to J. G. Mitchell and D. P. Moore; its formula is:

$$X_n = (X_{n-24} + X_{n-55}) \bmod \text{MAX0}$$

It is easily expressed in C; see Listing Four (page 27). It uses a 55-integer array initialized with numbers not all of which are even; usually these 55 seeds are drawn from a simpler QRNG (Talley-Metropolis will do). The constants 24 and 55 are not arbitrary; they were chosen because they produce a period of length 2^{55}. Nor

Dr. Dobb's Clinic (Text begins on page 20)

Listing One

```
rem recurse.bat called with %1 %2 %3 %4 %5 %6 %7 %8 %9
if (%1)==() exit
prompt command level %1$g
command /c recurse %2 %3 %4 %5 %6 %7 %8 %9
```

End Listing One

Listing Two

```
; Talley/Metropolis routines for 8086.  Consumer Warning: This
; generator is experimental and lacks a sound mathematical
; basis or statistical validation -- USE AT YOUR OWN RISK.
;
; Rand0: return a 16-bit quasi-random number in AX.
;       Preserves all registers except AX, flags.
Rand0:
        push    dx
        mov     dx,Xminus2
        mov     ax,Xminus1
        mov     Xminus2,ax
        add     ax,dx
        rol     ax,1
        mov     Xminus1,ax
        pop     dx
        ret
;
; RandI: return in AX a number in 0..CX-1.
;       Preserves all except AX, flags.
RandI:
        push    dx
        call    Rand0   ; ax = rand0()
        mul     cx      ; dxax = u*rand0()
        mov     ax,dx   ; ax = u*rand0()/MAX0
        pop     dx
        test    ah,80h
        jz      RandIx
        inc     ax      ; ax = u*rand0()/MAX
RandIx: ret
```

End Listing Two

Listing Three

```
; Talley/Metropolis routines for Z80.  Consumer Warning: This
; generator is experimental and lacks a sound mathematical
; basis or statistical validation -- USE AT YOUR OWN RISK.
;
; Rand0: return a 16-bit quasi-random number in HL.
;       Preserves all except HL, flags.
Rand0:
        push    de
        ld      de,(Xminus2)
        ld      hl,(Xminus1)
        ld      (Xminus2),hl
        add     hl,de
        add     hl,hl
        jr      nc,Rand0q
        inc     hl
Rand0q: ld      (Xminus1),hl
        pop     de
        ret
;
; RandI: return in HL a number in 0..BC-1.
;       Preserves all except HL, flags.
```

(Continued on page 26)

NASA has one shot to Jupiter. They'll go with dBASE III.

dBASE III and Ashton-Tate are trademarks of Ashton-Tate. ©Ashton-Tate 1985. All rights reserved.

NASA has only one shot at Jupiter with Project Galileo, so there's no room for error. The purpose of the project is nothing less than to find out the origin of the solar system and seek the answer to the nature and origin of life itself.

Galileo's success is all in the timing, and that's where dBASE III™ from Ashton-Tate™ takes control. dBASE III tracks the details of the sequence of launch events.

dBASE III was easy to bring on board. It's powerful (one billion records) and has a built in programming language that has been taught to speak "Galileo." dBASE III deals with the complex test details for Galileo as easily as it will deal with the complex details of your business for you.

When you've only got one shot at getting something right, you need the most powerful and popular data management system on Earth . . . or Jupiter.

For a dealer near you, call (800) 437-4329, ext. 2333. In Colorado call (303) 799-4900, ext. 2333.

Karen Boyle,
Data Programming Coordinator, Project Galileo

dBASE III. The data management standard.

ASHTON·TATE
SOFTWARE

Circle **no. 5** on reader service card.

are they unique; Knuth gives a table of other good pairs. Use 27 and 98 if you think you need a period of 2^{98}.

Random 8086

The routines in 8086 assembly language may be found in Listing Two (page 24). The seeding routine is omitted, as are details of segment definitions. Pay close attention to the consumer warning in the comments. It was Michael Barr who spotted the elegant equivalence between the C statements

```
c *= 2; if (c>MAX) c -= MAX;
```

and the single 8086 instruction

```
rol ax,1
```

Boy, if you ever see a C optimizer that can find that transformation, buy it.

Random Z80

The QRNG in Z80 assembly language is in Listing Three (page 24). The most complicated part of the Z80 (or any other 8-bit) version is the simulation of a 16-bit unsigned multiply. The one in Listing Three is a standard routine straight out of that classic work, *Z80 Programming for Logic Design* by Osborne, Kane, Rector, and Jacobson, published in the distant year of 1978 by Osborne and Associates and still one of our most useful references.

The only change we've made in Osborne's multiply is to use long jumps (JP opcodes) instead of short relative ones. A JP takes 10 clock cycles to do regardless, while a JR takes either 7 ticks to fall through or 12 to jump. Ordinarily, a JR is the opcode of choice because it is a byte shorter and, under the assumption that it will fall through and be taken with equal frequency, should consume half a clock cycle less on average.

However, if we can predict anything about the frequency with which a branch will be taken, we should make a choice between jump types. The last JP in Listing Three, for instance, will be taken 15 times and fall through but once. Clearly it ought to be a JP, since a JR would cost 30 more cycles.

The first of the three jumps should be a JR. It tests successive bits of the multiplier. In this application, those are bits of a quasi-random number, and the 50-50 assumption certainly ought to hold, indicating the use of JR. In a general-purpose multiply routine, we might expect that both multiplier and multiplicand will be positive binary numbers, but even if we expect only positive numbers less than 2^{15}—so that we expect the jump to be taken 9 of 16 times (i.e., always taken for the first two bits and taken for half of the other 14 bits)—the JR still squeaks out two-tenths of a clock cycle faster than a JP.

The middle jump is not so obvious, but because it will be executed in any manner on half or fewer of the iterations, it is also less important. In effect, we choose here between a 5-cycle JP or a 3.5/6 cycle JR. The jump will be taken when no carry results from adding the multiplier into the low word of the partial product. At first we thought that would be 3 out of 4 times, a clear win for JP over JR; on reflection, we ain't so sure. What do you think?

DDJ

(Listings begin on page 24)

¿C? ¡Sí!

If you're a C programmer (or want to be one), we speak your language. Subscribe to **The C Journal** today, and start increasing your productivity right away. We give you information you can **use** on **any** machine — IBM PC™, UNIX™-based, Macintosh™, or CP/M™ — micro, mini, or mainframe.

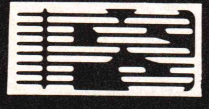
- in-depth reviews and feature articles — C compilers, editors, interpreters, function libraries, and books.
- hints and tips — help you work **better** and **faster**.
- interviews — with software entrepreneurs that **made it** — by using C.
- news and rumors — from the ANSI standards committee and the industry.

Limited Time Offer

Join our thousands of subscribers at the **Discount Rate** of only $18 for a full year (regularly $28)! Call us now at (201) 989-0570 for faster service — don't miss a single issue of **The C Journal!**

Please add $9 for overseas airmail.

Trademarks — CP/M: Digital Research Inc. IBM PC: IBM Corp. Macintosh: Apple Computer Corp. **The C Journal**: InfoPro Systems. UNIX: AT&T Bell Labs.



InfoPro Systems
3108 Route 10
Denville, NJ 07834
(201) 989-0570

MasterCard VISA

Circle no. 11 on reader service card.

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 190.**

Dr. Dobb's Clinic (Listing continued, text begins on page 20)

Listing Three

```
;       Requires Mul16 to do DEHL = BC*HL
RandI:
        call    Rand0   ; HL = rand0()
        push    de
        call    Mul16   ; DEHL = BC*rand0()
        ex      de,hl   ; HL = BC*rand0()/MAX0
        pop     de
        bit     0,h
        ret     z
        inc     hl      ; adjust to .../MAX
        ret

; a usable 16-to-32-bit multiply subroutine, producing
; DEHL as the 32-bit result of HL * BC, preserves BC, AF.
Mul16:
        push    af
        ld      a,16    ; loop count
        ex      de,hl   ; multiplier to DE
        ld      hl,0    ; clear partial product
Mul16A:
        ex      de,hl   ; next multiplier bit
        add     hl,hl   ; ..shifted to carry
        ex      de,hl   ; ..and 0-bit appended to hi prod
        push    af      ; (save m'ier bit)
        add     hl,hl   ; next product bit to carry
        jr      nc,Mul16B
        inc     de      ; adjust bit shifted earlier
Mul16B: pop     af      ; if m'ier bit was 1,
        jr      nc,Mul16C
        add     hl,bc   ; add multiplicand to product
        jp      nc,Mul16C
        inc     de      ; ..propogating carry
Mul16C: dec     a
        jp      nz,Mul16A
        pop     af
        ret
```

End Listing Three

Listing Four

```
/* Mitchell-Moore QRNG in C.  Presumes the existence of a */
/* simpler QRNG named r0rand() and its seeder, r0seed(x). */

extern void r0seed();
extern unsigned r0rand();

static unsigned xa[55];
static int j,k;

rseed(x)
unsigned x;
{
     r0seed(x); /* initialize rand0() predictably */
     for(j = 0; j < 55; ++j) xa[j] = r0rand();
     j = 23;
     k = 54;
}

rand()
{
unsigned x;
     x = xa[k] += xa[j];
     if (j==0) j = 55; --j;
     if (k==0) k = 55; --k;
     return(x);
}
```

End Listings

RGB COLOR BUSTER!! RGB

CONVERT THE RGB OUTPUT OF YOUR COLOR BOARD FOR THE IBM PC OR AN IBM PC COMPATIBLE TO COMPOSITE B/W WITH 16 LEVELS OF GRAY!

RUN **ALL APPLICATIONS** REQUIRING A COLOR MONITOR ON YOUR COMPOSITE BLACK AND WHITE MONOCHROME DISPLAY!

WITH THE ABILITY TO **ELIMINATE** OR **HIGHLIGHT** ANY COLOR GROUP, THE COLOR BUSTER OUTPERFORMS ANY OTHER COLOR TO B/W CONVERTER AVAILABLE.

BUY YOURS TODAY AT THIS SPECIAL INTRODUCTORY PRICE FOR A LIMITED TIME ONLY!

Only $69.95
($3.00 SHIPPING & HANDLING) (CALIF. RESIDENTS ADD 6.5% TAX)

STS Enterprises
5469 Arlene Way, Livermore, CA 94550
(415) 455-5358

Circle **no. 47** on reader service card.

Dr. Dobb's Journal

Subscription Problems?

No Problem!

Give us a call and we'll straighten it out. Today.

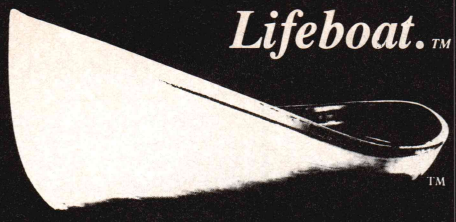
Outside California
CALL TOLL FREE: 800-321-3333
Inside California
CALL: 619-485-6535 or 6536

Sequitur™

- **As Relational As They Get (for micros)**
- **Word Processor Screen Editor Always in Operation**
- **New, Faster Version for DOS 2, 3, AT**

Golemics, Inc.
2600 10th St., Berkeley, CA 94710
(415) 486-8347

Circle **no. 45** on reader service card.



C is the language.
Lifeboat™ is the source.

Productivity Tools from the Leading Publisher of C Programs.™

The Lattice® C Compiler

The cornerstone of a program is its compiler; it can make the difference between a good program and a great one. The Lattice C compiler features:

- Full compatibility with Kernighan and Ritchie's standards
- Four memory model options for control and versatility
- Automatic sensing and use of the 8087 math chip
- Choose from the widest selection of add-on options
- Renowned for speed and code quality
- Superior quality documentation

"Lattice C produces remarkable code. . .the documentation sets such a high standard that others don't even come close. . .in the top category for its quick compilation and execution time and consistent reliability."

Byte Magazine

Lattice Library source code also available.

Language Utilities

Pfix 86/Pfix 86 Plus — dynamic and symbolic debuggers respectively, these provide multiple-window debugging with breakpointing capability.
Plink 86 — a two-pass overlay linkage editor that helps solve memory problems.
Text Management Utilities — includes GREP (searches files for patterns), DIFF (differential text file comparator), and more.
LMK (UNIX "make") — automates the construction of large multi-module products.
Curses — lets you write programs with full screen output transportable among all UNIX, XENIX and PC-DOS systems without changing your source code.
BASTOC — translates MBASIC or CBASIC source code directly to Lattice C source code.
C Cross Reference Generator — examines your C source modules and produces a listing of each symbol and where it is referenced.

Editors

Pmate — a customizable full screen text editor featuring its own powerful macro command language.
ES/P for C — C program entry with automatic syntax checking and formatting.
VEDIT — an easy-to-use word processor for use with V-PRINT.
V-PRINT — a print formatting companion for VEDIT.
CVUE — a full-screen editor that offers an easy way to use command structure.
EMACS — a full screen multi window text editor.
Fast/C — speeds up the cycle of edit-compile-debug-edit-recompile.

Graphics and Screen Design

HALO — one of the industry's standard graphics development packages. Over 150 graphics commands including line, arc, box, circle and ellipse primitives. The **10 Fontpack** is also available.
Panel — a screen formatter and data entry aid.
Lattice Window — a library of subroutines allowing design of windows.

Functions

C-Food Smorgasbord — a tasty selection of utility functions for Lattice C programmers; includes a binary coded decimal arithmetic package, level 0 I/O functions, a Terminal Independence Package, and more.
Float-87 — supports the 8087 math chip to boost the speed of floating-point calculations.
The Greenleaf Functions — a comprehensive library of over 200 routines.
The Greenleaf Comm Library — an easy-to-use asynchronous communications library.
C Power Packs — sets of functions useful for a wide variety of applications.
BASIC C — This library is a simple bridge from IBM BASIC to C.

Database Record Managers

Phact — a database record manager library of C language functions, used in the creation and manipulation of large and small databases.
Btrieve — a sophisticated file management system designed for developing applications under PC-DOS. Data can be instantly retrieved by key value.
FABS — a Fast Access Btree Structure function library designed for rapid, keyed access to data files using multipath structures.
Autosort — a fast sort/merge utility.
Lattice dB-C ISAM — a library of C functions that enables you to create and access dBase format database files.

Cross-Compilers

For programmers active in both micro and mini environments we provide advanced cross-compilers which product Intel 8086 object modules. All were developed to be as functional — and reliable — as the native compilers. They are available for the following systems:

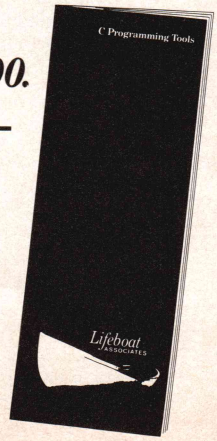
VAX/VMS, VAX/UNIX, 68K/UNIX-S, 68K/UNIX-L

Also, we have available:
Z80 Cross-Compiler for MS- and PC-DOS — produces Z80 object modules in the Microsoft relocatable format.

New Products

Run/C — finally, a C interpreter for all levels of C Programmers.
C Sprite — a symbolic debugger with breakpoint capability.

Call LIFEBOAT: 1-800-847-7078. In NY, 1-212-860-0300.



YES! Please rush me the latest FREE Lifeboat™ catalog of C products.

*Name*_____ *Title*_____

*Company Name*_____ *Business Phone*_____

*Address*_____

Please check one of the following categories:

☐ *Dealer/Distributor* ☐ *End User* ☐ *Other* _____

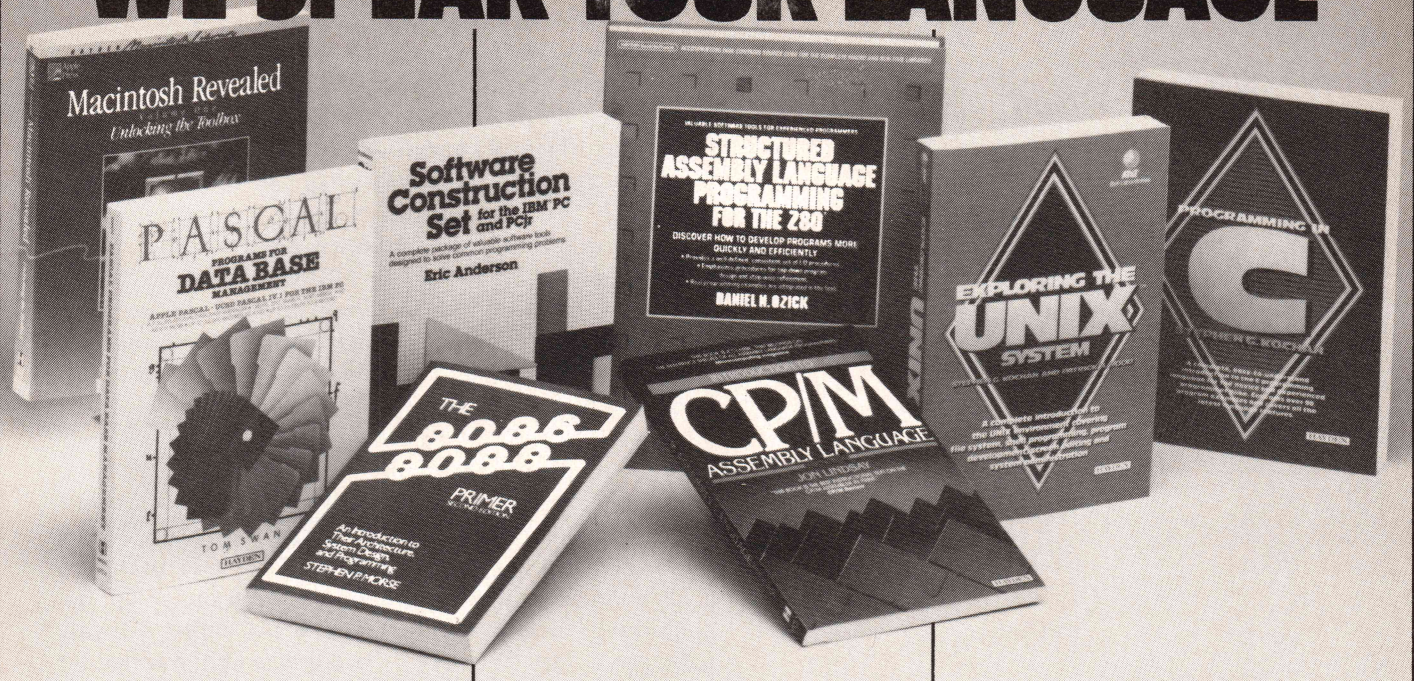
Return Coupon to: Lifeboat™ Associates
1651 Third Avenue, New York, NY 10128

© 1985 Lifeboat Associates

DD

Circle no. 125 on reader service card.

WE SPEAK YOUR LANGUAGE



Structured Assembly Language Programming for the Z80

By DANIEL N. OZICK. "Mr. Ozick is not only a fine programmer but also a solid technical writer. He knows his subject inside out and backwards. This is a superb piece of documentation."—*Microsystems.* (The book is accompanied by an 8½" disk containing source code for the complete macro and run-time libraries.) Book/Software: #7299-6; $49.95

Introduction to CP/M Assembly Language

Third Edition

By JON LINDSAY. "This book is the best instructional text on the CP/M assembler in print."—*CP/M Review.* "This book is a classic that belongs on the reference shelves of all assembly language programmers."—*Microcomputing.* #5210-3; $17.95

Programming in C

By STEPHEN G. KOCHAN. Written by one of AT&T's in-house programming instructors, this book contains over 90 program examples and covers all the latest language features. "Kochan's book, simply, is the BEST introduction to C."—*UNIX Review.* #6261-3; $21.95

Pascal Programs for Data Base Management

By TOM SWAN. Create, edit, search, sort, merge, and much more—here's the best bargain in data base management to date! Contains the source code for a complete relational data base system—ready to run—in Apple Pascal and UCSD Pascal IV.1 for the IBM PC. Book: #6272-9; $19.95/ Book/Software (Apple): #7272-4; $49.95/ Book/Software (IBM): #7273-2; $39.95

Software Construction Set for the IBM PC and PCjr

By ERIC ANDERSON. Put these IBM BASIC examples to work and see how fast, efficient, reliable, and easy to use your programs will be! Much more than a collection of specialized subroutines, this is a complete package of valuable software tools for the programmer. #6353-9/ $18.95

Exploring the UNIX System

By STEPHEN G. KOCHAN and PATRICK H. WOOD. An indispensable hands-on guide produced with AT&T Bell Laboratories' support, this complete introduction to the UNIX environment covers the file system, shell programming, program development, screen editing, and system administration. #6268-0; $21.95

Macintosh Revealed An Apple Press Book

Volume One: Unlocking the Toolbox
Volume Two: Programming with the Toolbox

By STEPHEN CHERNICOFF. One of the most eagerly anticipated books of the Mac era, this two-volume set reveals and explains all of the major ROM routines of the Macintosh Toolbox and puts them at your command. *Volume One* covers the basic elements of the operating system, focusing on the Finder and QuickDraw graphic routines. *Volume Two* moves on to higher level constructs of the Toolbox that will enable you to create your own programs that adhere fully to Apple Computer's Macintosh User Interface Guidelines. Volume One: #6551-5; $24.95/Volume Two: #6561-2; $29.95

The 8086/8088 Primer

An Introduction to Their Architecture, System Design, and Programming
Second Edition

By STEPHEN P. MORSE. Written by the person responsible for the design of the 8086 microprocessor, this revised edition has been updated to provide professionals with a thorough introduction to Intel's 8086 and 8088 microprocessors. In addition to chapters on a low-level programming language, ASM-86, and a high-level language, PL/M-86, a new chapter examines the Pascal language. #6255-9; $18.95

At bookstores and computer dealers or order direct today:
***Call 800-631-0856** (201-393-6300 in New Jersey). MasterCard and VISA accepted.*

Z80 is a registered trademark of Zilog. UNIX is a trademark of AT&T Bell Laboratories.
CP/M is a trademark of Digital Research. IBM PC and PCjr are registered trademarks of International Business Machines Corp.
Macintosh is a trademark licensed to Apple Computer, Inc.

HAYDEN BOOK COMPANY

10 Mulholland Drive, Hasbrouck Heights, NJ 07604
In Canada: Holt, Rinehart & Winston, 55 Horner Avenue, Toronto, Ontario M8Z #4X6

Circle **no. 53** on reader service card.

C Compilers for MSDOS

by the C Special Interest Group
of PicoNet and the Silicon Valley
Computer Society under the
direction of Richard Relph

Our reviewers found that some of the fastest compilers overall were relatively slow in doing screen I/O, and uncovered order-of-magnitude differences in execution time and generated-code size.

What would you consider the ideal credentials for a reviewer of computer language compilers? Would you want a software designer, or perhaps even a compiler designer? Should the reviewer be a professional who earns his or her living at the keyboard? Or should the review be conducted by the weekend programmer who really understands the value of user-friendly software as well as the frustration of user-hostile documentation?

The reviewer for this project is all of the above. This report on 13 C compilers for MSDOS was conducted by the joint C Special Interest Group (C-SIG) of PicoNet (a non-profit, general microcomputer users' group) and the Silicon Valley Computer Society (SVCS), an IBM PC users' group.

This combined C-SIG is composed of a broad cross-section of microcomputer users. Some are language-compiler designers, some design and maintain systems software, still others make their living developing commercial software such as spreadsheet or data base packages. There are several novice programmers in the group, but most are seasoned programmers, if not expert software designers. The "reviewer," therefore, is well qualified for the job.

Background

The first C-SIG was organized in 1983 by a group of PicoNet members. Their purpose was to provide a forum for discussion and study for those interested in the C programming language. In the beginning, classes were organized, and after 10 or 12 sessions the language was covered fairly thoroughly. The classes brought all members of the group to a common basic level of understanding of C.

The group soon graduated to developing software utilities: those tools that are commonly found on the larger systems for which C was originally developed, but that are not commonly available for the microcomputer environment. The C-SIG also organized group purchases of compilers, other software packages, and hardware at discounts, so that all members could own a quality C compiler, participate in the various group activities, and contribute to group projects.

Invariably, at the monthly meetings someone would ask what was the best current compiler (or version thereof), opening a debate over which compilers were the fastest, which produced the optimum code, which conformed most to "standard" C. We soon recognized from these discussions that the group was somewhat inbred; that is, among the 20 or 30 members, only two or three C compilers were represented (partly due to group purchases). Then, near the beginning of 1985, the PicoNet and SVCS C-SIGs merged.

SVCS was made up of IBM PC users. PicoNet was traditionally a CP/M group, but a large number of the original C-SIG members had recently acquired MSDOS-based PCs through a group purchase. Out of the group's new interest in MSDOS came increased interest in C compilers that ran under MSDOS. We soon discovered that the number of compilers available for the PC environment

was large indeed, and we quickly identified some 24 implementations. The questions of which was fastest, which produced the tightest code, and which was closest to the standard continued to come up, but we learned that no one had the answers. We decided to find them.

Given the magnitude of the task, the breadth of the group, and the number of individuals involved in the project, responsibilities had to be divided among the participating members. Each joined a group that was assigned a specific task critical to the overall project. The groups had responsibilities in the following areas:

- Standards: responsible for the evaluation of compilers' and libraries' adherence to the standard C language as defined by Kernighan & Ritchie (K&R) in *The C Programming Language*. Also responsible for qualifying all benchmarks written or acquired to measure compiler performance for universal compatibility and for assessing each compiler's ease of use (Richard Relph, Don Lawrence, Andi Pisiali, Stan Peters).
- Benchmarks: responsible for writing and/or acquiring C compiler benchmark programs and for conducting benchmark tests (Fred Viles, Steve Mahn, Scott Thomas, George Giomousis, Bill Overstreet, Fletcher Johnson).
- Documentation: responsible for reviewing documentation for clarity, completeness, style, and presentation. Also responsible for evaluating customer service for responsiveness and technical proficiency (Chuck Rulofson, Don Hockler, Bob Stephan, Peter Vutz).
- Writers: to put it all together (Irwin Diehl, Don Dodge).

That's the background—who we are, what we did, why we did it, and how. Oh yes, we should mention the "where." We've been most fortunate to have been provided a regular meeting place by ZeroOne Systems (formerly Technology Development of California). ZeroOne has been a most gracious host for the regular monthly meetings and the special weekend benchmark sessions. Their continuing support is appreciated.

Benchmarks

The benchmark group was charged with measuring the performance of the code produced by the various compilers. In the course of several meetings, we decided on our general approach and on the specific benchmark functions to be produced. We first discussed the possibility of producing a real-world benchmark program, one that would perform some useful function using an appropriate mix of language and library features. This plan, although attractive, was problematic since we could not agree on an appropriate mix. We settled on a "surgical" approach: many small functions, each intended to test a single feature of the compiler or its library.

We started by drawing up a list of every aspect of compiler and library performance that any one of us had ever wanted to see in other compiler reviews; not surprisingly, the list was enormous and impossible. In the interest of completing this project at all, we pared the list down, ending with 30 separate benchmarks to test performance

New C Books from Plum Hall

Efficient C (Thomas Plum and Jim Brodie, 1985) provides a small suite of C functions into which the reader can "plug in" any C statement and determine how many microseconds of CPU time it takes to execute. Expanding upon this technique, the book presents tables of CPU time and code space for C operators, control structures, and function calls. These allow the reader to make fairly accurate estimates of the resources that a program will take, without resorting to assembler listings. The book discusses optimization techniques performed automatically by several compilers, as well as those techniques which can be effectively used by the programmer. (Plum and Brodie are respectively Vice-Chair and Chair of the ANSI committee X3J11 which is standardizing C language.)

Reliable Data Structures in C (Thomas Plum, 1985) is an intermediate-level book that picks up where introductory books leave off. It describes techniques for a "no surprises" usage of pointers, structures, and files. Standard data structure techniques like stacks, queues, and trees are presented using reliability techniques, along with a complete menu-and-forms screen generator. Relevant information about the draft ANSI standard for C is presented to allow programmers to write compatibly with existing and forthcoming compiler and interpreter environments.

Learning to Program in C (Thomas Plum, 1983) presents the fundamentals of C. Presupposing only an acquaintance with computers, it covers C up through the basics of pointers and structures.

C Programming Guidelines (Thomas Plum, 1984) provides a style standard for projects working in C language. Arranged in "manual-page" reference format, it gives rules for using variables, data types, operators, expressions, statements, functions, files, libraries, and documentation.

Each book is $25.00 (plus 6% within NJ), $30.00 outside USA (airmail). We ship within two days when ordered by phone or mail with credit card, purchase order, or check.

Please send N copies of
__ Efficient C
__ Reliable Data Structures in C
__ Learning to Program in C
__ C Programming Guidelines

Plum Hall Inc
1 Spruce Av
Cardiff NJ 08232
609-927-3770

NAME _____
COMPANY _____
ADDRESS _____
CITY _____ STATE ____ ZIP ____

__ Check __ Mastercard __ Visa __ American Express

CARD NO _____ EXPIRATION _____
SIGNATURE _____

Circle **no. 70** on reader service card.

in the areas of code size; time to compile and link; storage class support; math; array and pointer referencing; function calling; optimization; screen and disk I/O; and general performance. We ran some of these functions for different memory models and some with the 8087 options available.

To get accurate and consistent results, we used a common format for all the execution-time benchmarks. Each benchmark was written as a function that accepted a loop count and returned the measured time in tenths of seconds. We wrote a pair of functions to measure an interval of time using the system clock, avoiding the problems inherent in the stopwatch approach. All compilers except Digital Research's provided a library function giving access to DOS interrupts, so the timer functions could be written easily in C for all but DRI. (So we wrote an assembly routine.) We then verified that the overhead in the timer functions was negligible for all compilers. The skeleton for all benchmark functions is in the Figure (below).

We adjusted each loop count to force the benchmark into a reasonable time frame (about 1 minute), so as to minimize the percent error in the timer functions. Although the system-clock interrupt returns a measurement of the time down to hundredths of a second, the clock is actually updated only 18 times per second, so the time reported for a given function can vary by as much as 0.2 seconds.

Except for the 8087 benchmarks, all the tests were performed on a Leading Edge PC. running MSDOS 2.11 and equipped with 640K RAM and a 10 Mb hard disk. These machines are close PC-compatibles running at a clock rate of 7.16 MHz; consequently, CPU-bound benchmarks will take 50 percent longer on an IBM PC. The Norton Utilities (V3) confirm this performance factor.

We selected the Leading Edge PC because we had made a group buy of several identically configured systems. We ran the compile-time and link-time benchmarks on the same machine in two different ways: using the hard disk, with care taken to ensure that files were read and written to the same location for each compiler, and on a 512K JDrive RAM disk. We ran the disk I/O benchmarks on the hard disk, again ensuring that the same disk locations were used. Before you moan "but I only have a plain IBM PC with floppies," remember that the goal here was to compare the compilers, not the machines. The ratio of the timings should be the same anywhere.

```
benchfn(loop)
int loop;
{
    /* Initialization */

    time_0( ); /* Start clock */
    for ( ; loop ; loop-- )
    {
        /* Code to be timed */
    }
    return(time_n( )); /* As sec/10 */
}
```

Figure

Benchmark Design

So much for background. Now we will discuss the benchmark functions, grouped by aspect to be tested. We have listings for all but the compile- and link-time programs, which were not actually executed. For a diskette (5¼-inch MSDOS) with the listings send a check for $6.00 payable to M&T Publishing to: *DDJ* Compiler Review, 2464 Embarcadero Way, Palo Alto, CA 94303.

Code Size

We wanted to get an idea of the granularity of the library and the minimum .EXE file size that could be produced. Granularity, for those not familiar with the term, refers to how finely the library is structured. A highly granular library allows the bare minimum of object code to be linked for each function. Four programs were written for these tests:

- minmain.c—Main program with no executable code.
- minputs.c—Main program with one puts( ) call.
- minprtf.c—Main program with one printf( ) call.
- minfio.c—Main program with calls on fopen( ), fgets( ), fputs( ), and fclose( ).

We compiled and linked, but did not execute, each program. We were especially interested in minprtf.c, since using printf( ) usually means dragging in the whole floating-point package. Some compilers provided an option to link with an integer-only version of printf( ). In those cases we linked minprtf.c twice, once with the integer-only and once for the full floating-point versions of printf( ). See Table 1 (page 34) for the results.

Clocking Compile and Link Time

Since most of the performance benchmarks are very short and not representative of typical programs, we used separate programs ranging from 1 to 1000 lines for this test. They were:

- doc1.c—A one-liner, to show the minimum overhead.
- doc10.c—A 10-liner, about the smallest module you'd ever compile.
- doc100.c—This 100-liner is the most significant since it represents a typical size of source file.
- doc1000.c—A 1000-line program. Basic overhead should be minimal here.

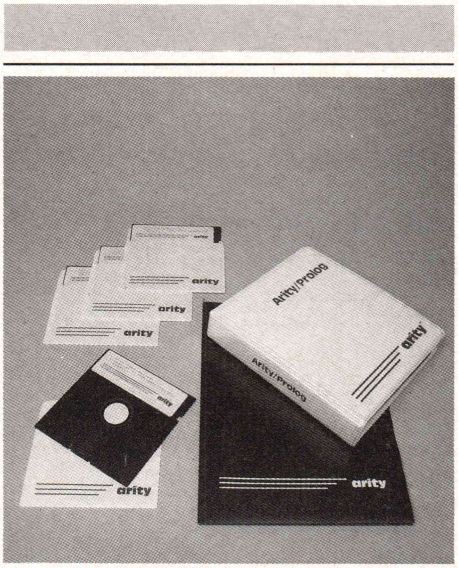
Doc100.c was produced from a preexisting program belonging to one of the group. Doc1000.c is a stripped-down version of the DIFF program available from DECUS (Digital Equipment Corp. Users' Society).

Since modular compilation tends to be the rule with C programming, we timed the compile and link steps separately. The compile step included all phases of the compila-

More Power Than You Thought Possible

Arity offers the first serious implementation of Prolog for IBM personal computers. Arity/Prolog is a powerful, highly optimized, and extended version of the logic programming language Prolog. Imagine building software applications with a language that solves problems through deduction and logical inference. The task of creating complex programs is much faster and easier, resulting in lower development costs. Arity/Prolog is now in use in a wide range of applications in industry, business, research, and education. The solution—the *Arity/Prolog Interpreter:*

- Source level debugger
- Virtual databases, each with a workspace of 16 megabytes
- Floating-point arithmetic
- String support for efficient text handling



- Interface to assembly language and 'C'
- Text screen manipulation
- Integrated programming shell to MSDOS
- Comprehensive set of evaluable predicates
- Definite clause grammar support

Arity/Prolog Interpreter $495.00

Arity also offers the *Arity/Prolog Compiler and Interpreter,* a sophisticated development environment for building AI applications. Essential for producing fast, serious production code.

Arity/Prolog Compiler and Interpreter $1950.00

The *Arity/Prolog Demo Disk* is available for $19.95. ■ Arity/Prolog products run on the IBM PC, XT, AT, and all IBM compatibles. ■ To order, call (617) 371-2422 or use the order form below.

arity corporation 358 Baker Avenue, Concord, MA 01742

Name

Organization

Address

Quantity	Product	Unit Price	Total Price
	Arity/Prolog Compiler & Interpreter	$1950.00	
	Arity/Prolog Interpreter	$ 495.00	
	Arity/Prolog Demo Disk	$ 19.95	
		Subtotal	
		MA residents add 5% sales tax	
		Total Amount	

☐ Enclosed is a check or money order to Arity Corporation

☐ Please bill my ☐ Mastercard ☐ Visa ☐ American Express

Account #

Valid ____/____ to ____/____ signature

☐ Please send me more information about Arity and Arity/Prolog

arity 358 Baker Avenue, Concord, MA 01742

M-AD-05

Circle **no. 16** on reader service card.

DeSmet C

MACINTOSH™ DEVELOPMENT PACKAGE

$150 Includes Shipping

Runs on 128K and 512K Macintosh
- Produces FINDER/SHELL applications
- Dynamic OVERLAY support

Full K&R C Compiler
- Native Code Compiler
- In-line **asm** directive
- IEEE S/W Floating Point

Assembler, Linker, and Librarian

Machine Code Debugger

Source Code Editor

"SHELL" Interface
- Environmental Variables
- Wild-Card Expansion
- Many Built-in Functions
- Command History
- Runs Any Application

>120 Function STDIO Library

>450 Function Macintosh ROM Library

360 Page Manual

RAM Disk

Macintosh is a trademark licensed to Apple Computer, Inc.

☐ Please Send Information.
☐ Send Macintosh Development Package

Check # _____ Enclosed

SHIP TO:

ZIP

CWARE CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface. California residents add sales tax. Canada shipping add $5, elsewhere add $15. Checks must be on U.S. Bank and in U.S. Dollars. Call 9 a.m.–1 p.m. to CHARGE by **VISA/MC/AMEX.**

Street Address: 505 W. Olive, #767, Sunnyvale, CA 94086

Circle **no. 18** on reader service card.

Vendor	minmain	minputs	minprtf	minfio	DOC1	DOC10	DOC100	DOC100
Aztec (Manx)	1856	3136	**4336**	**3952**	1600	4416	**6784**	**10928**
Control C	6698	6714	6714	7846	6698	6794	12198	14352
C Systems	12032	12032	19456	12544	11934	19440	25018	28464
Computer Innovations	4352	4480	9344	5632	9408	9408	12789	16956
Datalight	3386	3603	5644	**3927**	2020	7732	12148	NA
DeSmet	1536	1536	6656	8192	1536	6656	14848	19456
Digital Research	12800	13440	18816	15616	12800	18944	24704	28416
EcoSoft	**692**	**2452**	7616	4026	**708**	**770**	11528	NA
Lattice	10278	10306	11702	10812	10294	11786	14654	19688
Mark Williams	6698	6714	6714	7846	6698	6794	12198	14352
Microsoft	1912	4296	5750	5896	1928	5802	9090	14684
Software Toolworks	3918	5286	6850	7422	8556	10964	14180	22192
Wizard	**716**	**2440**	8168	4278	**732**	8236	12234	16336

Table 1
.EXE File Size

Vendor	DOC1		DOC10		DOC100		DOC1000	
	compile	link	compile	link	compile	link	compile	link
Aztec (Manx)	**2**	**2**	3.5	**3**	16.5	**4**	103	**5.5**
Control C	2	12	**3**	12	12	19	69	21
C Systems	6	11	9	14	39.5	17.5	233	19
Computer Innovations	6	10	7	14	19	16	105	20
Datalight	2.5	3	3	5.5	12	9	NA	NA
DeSmet	3	3.5	**3**	4	**6**	**4.5**	**28**	**5**
Digital Research	3	16	4	20	17	25	120	29
EcoSoft	3	3	4	13	13	18.5	NA	NA
Lattice	2.5	9.5	4	10	18.5	11.5	126	15.5
Mark Williams	**2**	12	**3**	12	12	19	69	21
Microsoft	4	3	6	5.5	20	7	131.5	9.5
Software Toolworks	3	11	6	13	28	15	243.5	20
Wizard	3	**2**	**3**	7	10	9	54	12.5

Table 2a
Compile and Link Times: RAM Compilation

Vendor	DOC1		DOC10		DOC100		DOC1000	
	compile	link	compile	link	compile	link	compile	link
Aztec (Manx)	**8**	**5.6**	9.5	**8**	24.5	**10**	116	**12**
Control C	13	20	14	20	26.5	27.5	93	30
C Systems	17	18	20	22	52	25	255	27
Computer Innovations	21.5	15	22	20	37	23	128	28
Datalight	8.5	7	**9**	11.5	20.5	15	NA	NA
DeSmet	10	7	10	8.5	**14.5**	**10**	**43**	**11**
Digital Research	13	24	14	29	28	36	135	40
EcoSoft	14	7	15	20.5	27	26	NA	NA
Lattice	9	15.5	10	16	27.5	18	149	22.5
Mark Williams	13	20	14	20	26.5	27.5	93	30
Microsoft	19	8	21.5	13	38	15	159.5	20
Software Toolworks	9	18	11.5	21	35	24	254	30
Wizard	15	5.5	17	13	27	16	78	20

Table 2b
Compiler and Link Times: Hard-disk Compilation

Vendor	Overall performance and misc.						
	looptst	optim	fibtest	sieve	rsieve	psieve	isort
Aztec (Manx)	**37.8**	35.7	56.2	**99.0**	58.0	53.3	42.0
Control C	62.2	40.9	63.0	103.4	59.1	54.8	60.0
C Systems	100.8	73.1	69.7	173.8	132.6	190.7	51.5
Computer Innovations	66.1	39.8	54.0	117.5	117.4	119.3	49.0
Datalight	63.0	35.3	**49.8**	**99.9**	100.0	101.3	**42.6**
DeSmet	63.0	46.4	52.9	108.2	108.4	103.4	53.8
Digital Research	58.8	66.9	51.7	104.8	58.8	57.0	49.9
EcoSoft	63.0	50.7	52.0	115.3	115.4	138.1	85.7
Lattice	63.0	**25.0**	70.0	100.3	100.2	96.9	77.2
Mark Williams	61.8	40.8	63.0	103.3	59.0	54.7	59.7
Microsoft	58.8	59.0	59.0	99.4	**54.9**	**50.0**	44.3
Software Toolworks	74.2	31.8	56.4	133.6	110.3	176.8	123.6
Wizard	58.8	68.8	52.6	101.0	55.7	51.1	60.0

Table 3a
Execution Times

We're getting hardnosed at Softway.

From now on MATIS™ is only $49.95!

(MATIS, the complete User Interface development tool has been selling for $150.)

Why the radical price cut?

We decided after looking over the competition that MATIS had so many advantages it should be made available to more programmers. We decided to compete aggressively so you could easily afford to have MATIS in **your** bag of tricks. We hear from MATIS users in the USA and France that it is a truly loveable product. Sooo...we're running this big ad to promote our new low price.

MATIS windows are beautiful.

Display any portion of any screens you create at any point in your program. Scroll in any direction manually with cursor keys...or automatically.

And the screens are HUMUNGEOUS!

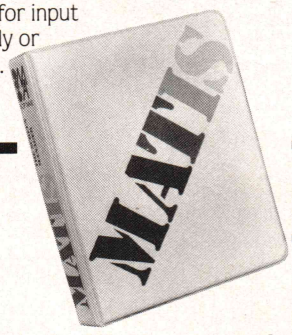
MATIS screens can be just about as big as you want...up to 65,534 rows by 65,534 columns! The number of screens is only limited by available memory.

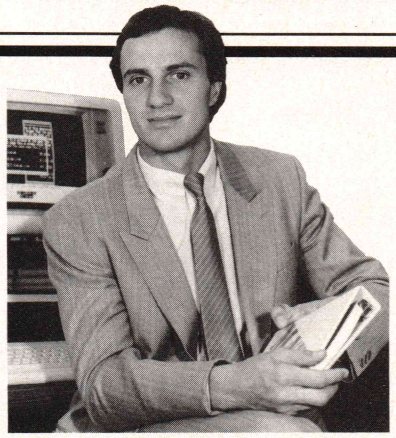
Print big MATIS screens directly.

One command sends your screens to your printer with no need to program special routines when your virtual screen is bigger than your terminal screen.

User input fields are a snap.

Creating fields for data entry is easy with no limit to size or number by screen. Request for input separately or in blocks.





Denis Moran
President, Softway, Inc.

Control your keyboard with MATIS.

It keeps track of keys that are pressed during the execution of your program and lets you assign specific functions to selected keys.

Control that screen too!

MATIS is extremely versatile and flexible when it comes to controlling lines, columns, fields, and text. They can be modified, transferred, displayed or moved with a single command. All video attributes are supported: color, reverse video, blinking...you name it, you got it.

Want an interactive screen builder?

You've got it with MATIS. It's called "MATPAGE"™ and it lets you create and modify any of your screens in an interactive mode.

MATIS adds over 70 routines to your program.

Written in Assembler, MATIS routines are fast and powerful giving your program improved efficiency and enhanced visual appeal, while they reduce its size and maintenance worries. And MATIS separates screen design from the core of your program.

MATIS is unique.

We don't think there's a single program that combines as many tools in one package as completely or as well as MATIS. It interfaces with Intepreted and Compiled BASIC (Microsoft), C (Lattice, Microsoft, Aztec), PASCAL (IBM, Microsoft) and ASSEMBLER. All you need is an IBM* PC/XT or true compatible under DOS, 128k or RAM, monochrome or color monitor.

You get an easy to follow no-frills manual and a 30-Day Money Back Guarantee.

Late News: MATIS/T™ for TURBO-PASCAL** only $29.95

An indispensable add-on at a dynamite price. What more can we say?

Denis Moran

™MATIS, MATIS/T, & MATPAGE are Trademarks of Softway, Inc.
*IBM is a Reg. Trademark of IBM Corp.
**Turbo Pascal is a Reg. Trademark of Borland International

Softway, Inc.

500 Sutter Street, Suite 222 H
San Francisco, California 94102
(415) 397-4666

24-Hour Credit Card Orders By Phone:
1 (800) 227-2400 EXT 989 In California: 1(800) 772-2666 EXT 989

Please ship the following at once. I understand there is a 30-day money back guarantee.

_____ Copies of MATIS at $49.95 plus $3 shpg. $_____

_____ Copies of MATIS/T for TURBO PASCAL at $29.95 plus $2 shpg. $_____

California residents, add 6½% sales tax $_____

☐ I like to read specs, so send me a folder. Total $_____

Name _____

Address (please no P.O. Box) _____

City/State/ZIP _____

Phone (_____) _____ Signature _____

Make payment by money order, check, or charge card ☐ VISA ☐ MASTER CARD

Number _____ Exp._____

Distributed in Europe By: MICRO APPLICATION SOFTWARE • 147 Avenue Paul Doumer, 92500 RUEIL MALMAISON FRANCE, Tel (1) 732.92.54

Circle **no. 92** on reader service card.

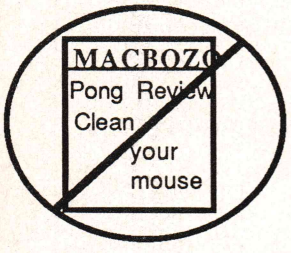
tion, including assembly, if any. The compile and link process was controlled by a batch file since we weren't interested in benchmarking the testers' typing speeds. Some compilers provided a control program that allowed the linker to be run automatically after the compile with a single command. In these compilers, we defeated that option and issued a second command to do the link so we could derive separate compile and link times. To produce the time estimate, we used the Norton Utilities timemark program, invoking it from the batch file at the appropriate points. See Tables 2a and 2b (page 34) for these results.

Testing Screen and Disk I/O

We wanted to test how fast the generated code could write to the console using puts( ) and printf( ), as well as how fast it could read and write to the disk. We wrote six functions to find out.

- The fillscr.c function used puts( ) to write a string of 1248 characters to the screen without scrolling. Scrolling was prevented by using a bare carriage return every 78th character of the string. The screen gradually fills because puts( ) adds its own newline at the end of the string, and the puts( ) call is repeated <loop> times. The screen was cleared by the DOS CLS command prior to running this test.
- The scroll.c function was almost the same as fillscr( ), except the string had a newline every 78th character so each puts( ) produced 16 lines of characters plus a blank line on the screen. So the time difference between scroll( ) and fillscr( ) measured the scrolling overhead.
- The prtf.c function produced the same number of characters and lines as scroll( ), except it used printf( ) instead of puts( ) and executed each basic format conversion several times. Comparing the time for prtf( ) to that for scroll( ) shows how efficiently the compiler's printf( ) function performs its conversion duties.
- The cpychr.c function copied a 10,000-byte file using fgetc( ) and fputc( ).
- The cpyblk.c function copied a 10,000-byte file using fread( ) and fwrite( ) to copy in 1024-byte blocks. The difference between the times for these functions reveals the advantage blocked I/O has in a particular compiler's library.
- The diskio.c function was designed to test random character I/O to a reasonably large data file. It used fseek( ), fgetc( ), and fputc( ) to randomly read and write individual characters in a previously created data file 240,000 bytes long.

We argued about whether to use fopen( ), fread( ), fwrite( ), and the like, rather than open( ), read( ), write( ), and other corresponding functions. The latter functions were provided with many of the compilers and might have produced faster times, but ultimately we decided to use the f functions, primarily on portability con-



NO FLUFF!

MacTutor, the Macintosh Programming Journal.
For the few of us who already know what a computer is for.

$24 per year US funds.
$30 for Canada.
$36 for Overseas.

Twelve issues per year of technical info on C, Assembly, Neon, Basic, Forth, Pascal and Lisp. Programming ON the Mac, FOR the Mac!

MacTutor
P.O. Box 846
Placentia, CA. 92670

Not convinced yet? Call (714) 993-9939 for a free sample.

Circle **no. 54** on reader service card.

Lattice presents
The dBC library

The new Lattice dBC Library commands great performances between C language programs and dBASE files.

With the dBC software tool kit, you can extend any existing dBASE-II or dBASE-III application, or write complete new applications. dBC provides 37 functions that easily add, update, delete, retrieve and organize records as well as their corresponding indexes. And up to 8 data and 8 index files can be opened and processed simultaneously.

The dBC Library accesses dBASE files under the MS-DOS, PC-DOS and UNIX environments.

dBASE is a trademark of Ashton-Tate
dBC is a trademark of Lattice

It operates with popular C compilers such as Lattice C, CI-C86 and DeSmet C88. No object license is required to re-sell products developed with the dBC Library.

Put the spotlight on your dBASE programming...order today!

PRICE: $250 per copy
($500 with source code)

ASK ABOUT OUR "TRADE UP TO LATTICE C POLICY"

For more information or to place your order contact:

LATTICE
P.O. Box 3072
Glen Ellyn, IL 60138
TWX 910-291-2190

Circle **no. 58** on reader service card.

siderations. The February 1985 preliminary draft of the proposed ANSI standard does not include open( ), et al., on the grounds that they are too closely tied to the Unix operating system and that they are redundant. We dodged the potentially sticky problem of conversion between newline and CR-LF by making sure that none of the data files contained these characters.

Testing String Library Functions

We wanted to test some library functions beyond the usual I/O functions. We were concerned, though, that many of the functions we could test would not be provided with all of the compilers, so we wound up with a single benchmark function, isort.c, to test strcmp( ) and strcpy( ). This function used strcmp( ) and strcpy( ) to perform a simple sort of an array of up to 1024 eight-character strings. (In this case, the <loop> parameter was used to control the number of elements to sort; it does not represent the number of sorts performed.) The isort.c test tells little about compiler performance per se, but it does give a clue as to how well the runtime library was written.

Math Tests

We wrote four functions to test basic arithmetic functions for various data types.

- The tint.c function includes 1500 add, 1600 subtract, 200 multiply, and 200 divide operations, <loop> times. Inner loop overhead amounts to <loop> * 100 test-for-zero and decrement operations. Since there are not many ways to perform 16-bit integer arithmetic on an 8088, we expected most of the compilers to perform about equally on this one.
- The tlong.c function was exactly the same as tint( ), except that the variables involved in the adds, subtracts, multiplies, and divides were all declared long instead of int. Significant differences between compilers were more likely here.
- The tfloat.c function tested floating-point arithmetic with 40 each adds, subtracts, and multiplies and 20 divides, all done <loop> times.
- The tdouble.c function was the same, except that the variables were declared double instead of float. For most compilers, tdouble( ) should be somewhat faster than tfloat( ) since K&R calls for all floats to be converted to double before being used in an expression. Tfloat( ) and tdouble( ) were run with whatever flavors of floating-point support (including 8087 options) were available for each compiler. The results of these tests appear in Table 3b (page 38) and Table 4 (page 43).

Array and Pointer Tests

We wrote two functions to explore how efficiently the compilers handled array- and pointer-referencing tasks.

A Professional Quality Z80/8080/8085 Disassembler

WHEN YOU NEED SOURCE FOR YOUR CODE
you need REVAS 3

REVAS interactively helps you:

Analyse your software for modification
disassemble files as large as 64K
Assign Real labels in the disassembly
Insert COMMENTS in the disassembly
Generate a Cross Reference (XREF) listing

A 60 page manual shows how the powerful **REVAS** command set gives you instant control over I/O to files, printer, or console; how to do a disassembly; and even how the disassembler works! You get on line help, your choice of assembler mnemonics, control of data interpretation, and calculation in any number base!

REVAS runs in Z80 CPM computers; is available on 8" SSSD (standard), RAINBOW, and other (ask) formats

Price: $90.00 (plus applicable tax), Manual only: $15.00

REVASCO
6032 Chariton Ave., Los Angeles, CA 90056
Voice: (213) 649-3575 Modem: (213) 670-9465

Circle **no. 80** on reader service card.

CP/MAC™

CP/MAC gives you CP/M-80 programs the way you want, and the way the Macintosh™ was meant to be used. They are on the same disks with your other Macintosh files. You can open and start them the same ways you open paintings or other Macintosh documents. And graphic controls and cues help you while they are running - including a continuous display of disk availability.

CP/MAC works with hard disks and RAM disks. With a 512k Mac, CP/MAC gives you Z80 emulation - not available anywhere else. And CP/MAC transfers programs and any other files from CP/M computers. CP/MAC is a quality product and shipping now. At a price you can afford - $135. Order by phone (714-953-8985) or mail to:

LOGIQUE
30100 Town Center Dr. "O" Suite 198
Laguna Niguel, Ca. 92677

MC/VISA or COD ok. Or ask your dealer.

8080 emulation with about 42k TPA on a 128k Macintosh. Z80 emulation with 63k TPA on a 512k Macintosh. CP/M 2.2 BDOS and BIOS calls. LST and PUN output to the printer, RDR and CON input from the keyboard, and CON outputs to the CP/MAC window. ADM-3A emulation with reverse video and line insert/delete.

Macintosh is a trademark licensed to Apple Computer, Inc. CP/M is a registered trademark of Digital Research, Inc. CP/MAC is a trademark of Logique.

Circle **no. 9** on reader service card.

The function array.c simply initialized a three-dimensional array of 1000 ints, and pointer.c did the same, but using a pointer to a pointer to a pointer to an int. The unnecessary extra levels of indirection were added to increase the amount of pointer dereferencing relative to the overhead; i.e., to increase the ratio of data to noise.

Function Call Tests

To test the overhead due to function call and return, we created a version (fibtest.c) of the fib benchmark from the August 1983 issue of *Byte* magazine. (This is a good test of function-call overhead because it's composed entirely of negligible arithmetic and recursive calls.) As with tint( ), the optimum code for function calls and returns is well known, so we did not expect to see much variation between compilers on this test, although significant differences would be interesting.

Storage Class Tests

We were curious about the efficiency of the code generated for variables of various storage classes and in particular about how well register variables were supported. To find out, we wrote three almost-identical functions: autotst.c, stattst.c, and regtest.c. Each contained four empty **for** loops. The only difference among the functions is that the four loop variables were declared with automatic, static, and register storage classes, respectively.

Looking For Optimization

We thought it would be a nice idea to try to determine just how extensive each compiler's optimization algorithms were. To that end we wrote optimize.c, a mishmash of statements that a compiler could reasonably be expected to optimize. The idea was that the more cases a compiler noticed and optimized, the better its performance on the test would be.

Assessing General Performance

The looptst.c function did nothing at all; it was just an empty loop. We included it so that we could get an idea of how much overhead was being introduced by our outer loop in each of the benchmarks. Looptst( )'s inner loop did nothing 10,000 times, so there were 5,000,000 total loop iterations.

We knew we could not get away with not running the infamous *Byte* sieve benchmark, so we decided to go overboard and produce three, starting with sieve.c, our version of the standard sieve benchmark as published in the January 1983 *Byte*. We added rsieve.c, identical to sieve( ) except that the two most frequently referenced variables were declared register. This is the way some vendors run the sieve when producing numbers for their advertising copy. And we wrote psieve.c, a pointerized version of the sieve. This function may not prove very useful in comparing compilers, but writing it was an interesting exercise. It showed us that it is not true that using pointers is always better than using array references. Psieve( ) is considerably less readable and performs only marginally better than rsieve( )—on some compilers it actually ran at a slower rate!

Vendor:	Math functions:				Array/ptr:
	tint	tlong	array	ptr	
Aztec (Manx)	**34.0**	125.4	**30.7**	**50.4**	
Control C	36.4	176.1	96.4	72.5	
C Systems	39.3	247.9	90.0	85.0	
Computer Innovations	36.9	215.8	95.1	71.9	
Datalight	35.4	**87.7**	61.4	65.2	
DeSmet	36.0	178.9	93.6	68.0	
Digital Research	**34.1**	808.4	96.8	68.0	
EcoSoft	36.0	298.8	91.7	68.0	
Lattice	36.3	192.8	60.2	66.8	
Mark Williams	36.7	175.4	96.4	72.5	
Microsoft	**33.5**	102.9	76.2	**50.4**	
Software Toolworks	59.0	N.A.	131.5	116.8	
Wizard	**34.0**	173.0	78.4	**50.4**	

Vendor:	Screen I/O:		Disk I/O:				Storage class support:		
	fillscr	scroll	prtf	cpyblk	cpychr	diskio	autost	stattst	regtst
Aztec (Manx)	50.5	57.8	61.7	98.2	**39.7**	62.8	**70.6**	**79.3**	**54.2**
Control C	**22.3**	**29.0**	37.6	104.2	59.5	47.6	72.3	81.1	**54.1**
C Systems	31.0	35.5	42.1	182.6	82.2	**26.7**	126.9	136.0	108.8
Computer Innovations	27.8	32.6	58.1	147.0	**42.0**	50.9	79.3	86.6	79.3
Datalight	49.9	54.7	63.9	123.6	53.3	N.A.	79.4	86.5	79.4
DeSmet	27.8	31.5	48.1	128.8	94.6	48.1	79.4	86.5	79.4
Digital Research	37.5	41.9	**35.9**	176.0	83.3	36.2	**70.5**	**79.3**	**52.2**
EcoSoft	31.3	37.8	40.8	N.A.	N.A.	N.A.	79.4	86.5	79.4
Lattice	62.4	67.5	71.0	129.0	63.5	66.1	79.3	86.5	79.4
Mark Williams	**22.3**	**29.0**	37.5	156.1	85.0	42.4	72.2	81.1	**54.0**
Microsoft	47.7	55.0	59.1	**73.0**	53.0	57.0	**70.5**	**79.3**	**52.2**
Software Toolworks	62.5	67.6	67.9	726.0	334.2	55.3	112.6	136.0	112.5
Wizard	49.7	54.7	65.5	139.7	60.4	55.5	**70.6**	**79.3**	**52.3**

Table 3b
Execution Times continued

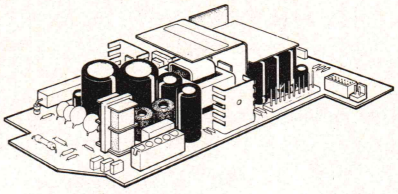
ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

Switching Power Supply!

+ 5VDC - 8 Amps
+12VDC - 5 Amps
-12VDC - 1 Amp
(81 WATTS MAX)



$29.95 ea.

4 FOR $99

ADD $2 EA. UPS

BRAND NEW UNITS, MFG. BY BOSCHERT FOR HEWLETT PACKARD! PERFECT FOR SMALL COMPUTER AND DISK DRIVE APPLICATIONS #XL81-5630. INPUT 110V/220V, 50/60 HZ. NOMINAL OUTPUTS: +5VDC @ 8A, +12VDC @ 5A, -12VDC @ 1A. TOTAL MAX OUTPUT. MUST BE LIMITED TO 81 WATTS TOTAL! (WITH PIN OUT SHEET.) ORIGINAL FACTORY BOXED.

64K S100 STATIC RAM

$139.00 KIT

NEW!

LOW POWER!

150 NS ADD $10

BLANK PC BOARD WITH DOCUMENTATION $49.95

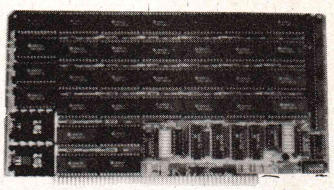
SUPPORT ICs + CAPS $17.50

FULL SOCKET SET $14.50

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED)

FOR 56K KIT $125

ASSEMBLED AND TESTED ADD $50



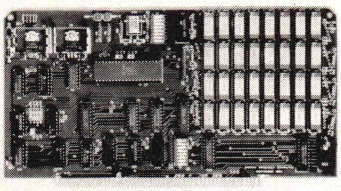
PRICE CUT!

FEATURES:
- Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- Fully supports IEEE 696 24 BIT Extended Addressing.
- 64K draws only approximately 500 MA.
- 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- 2716 EPROMs may be installed in any of top 48K.
- Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- BOARD may be partially populated as 56K.

256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



FEATURES:
- 256K on board, using + 5V 64K DRAMS.
- Uses new Intel 8203-1 LSI Memory Controller.
- Requires only 4 Dip Switch Selectable I/O Ports.
- Runs on 8080 or Z80 S100 machines.
- Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- Provisions for Battery back-up.
- Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
- The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) $69.95 (8203-1 INTEL $29.95)

(ADD $50 FOR A&T)

$169.00

#LS-100 (FULL 256K KIT)

64K SS-50 STATIC RAM

$119.00 (48K KIT)

NEW!

LOW POWER!

RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

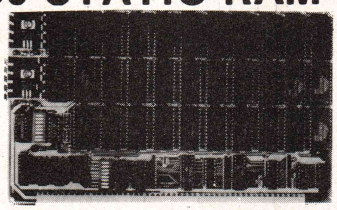
SUPPORT ICs + CAPS $18.00

FULL SOCKET SET $15.00

56K Kit $129

64K Kit $139

ASSEMBLED AND TESTED ADD $50



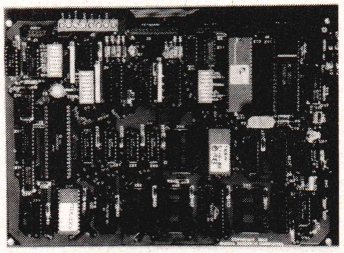
FEATURES:
- Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- Fully supports Extended Addressing.
- 64K draws only approximately 500 MA.
- 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- 2716 EPROMs may be installed anywhere on Board.
- Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- One Board supports both RAM and EPROM.
- RAM supports 2MHZ operation at no extra charge!
- Board may be partially populated in 16K increments.

THE NEW ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:
- Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- RS232 at 16 BAUD Rates from 75 to 19,200.
- 24 x 80 standard format (60 Hz).
- Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- Composite or Split Video.
- Any polarity of video or sync.
- Inverse Video Capability.
- Small Size: 6.5 x 9 inches.
- Upper & lower case with descenders.
- 7 x 9 Character Matrix.
- Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10



$89.95 #ZRT-80

(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM $49.95

SOURCE DISKETTE - ADD $10

SET OF 2 CRYSTALS - ADD $7.50

32K S100 EPROM/STATIC RAM

NEW! FOUR FUNCTION BOARD! **NEW!**

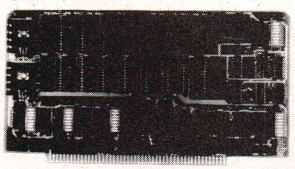
EPROM II FULL EPROM KIT $69.95

A&T EPROM ADD $35.00

BLANK PC BOARD WITH DATA $39.95

SUPPORT IC'S PLUS CAPS $16

FULL SOCKET SET $15



We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

FEATURES:
- This one board can be used in any one of four ways:
  - A. As a 32K 2716 EPROM Board
  - B. As a 32K 2732 EPROM Board (Using Every Other Socket)
  - C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
  - D. As a 32K Static RAM Board
- Uses New 2K x 8 (TMM2016 or HM6116) RAM's
- Fully Supports IEEE 696 Buss Standard (As Proposed)
- Supports 24 Bit Extended Adressing
- 200 NS (FAST!) RAM'S are standard on the RAM Kit
- Supports both Cromemco and North Star Bank Select
- Supports Phantom
- On Board wait State Generator
- Every 2K Block may be disabled
- Addressed as two separate 16K Blocks on any 64K Boundary
- Perfect for MP/M* Systems
- RAM Kit is very low power (300 MA typical)

PRICES SLASHED!

32K STATIC RAM KIT — $109.95

For RAM Kit A&T — Add $40

Digital Research Computers

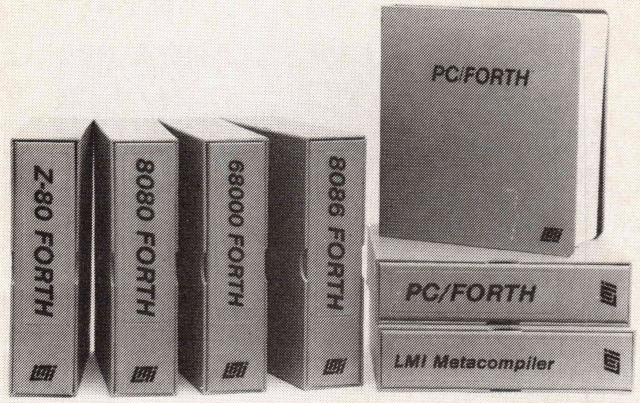
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add $3.00 postage. We pay balance. Orders under $15 add 75¢ handling. No. C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for for insurance.

*TM OF DIGITAL RESEARCH INC. (CALIF.)

WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

TOTAL CONTROL
with *LMI FORTH*™



For Programming Professionals:
an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, and 6502
- No license fee or royalty for compiled applications

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, D-7820 Titisee-Neustadt
UK: System Science Ltd., London EC1A 9JX
France: Micro-Sigma S.A.R.L., 75008 Paris
Japan: Southern Pacific Ltd., Yokohama 220
Australia: Wave-onic Associates, 6107 Wilson, W.A.

Circle **no. 55** on reader service card.

Testing Memory Models

All the benchmarks mentioned were compiled and run under the small-memory model. To see what kind of performance penalty you pay to use other models, we compiled and ran array( ), pointer( ), and psieve( ) with every other memory model option available with each compiler. In retrospect, we probably should have included fibtest( ) and maybe sieve( ) in the set.

Benchmark Results

You can see the results for all these benchmarks in Table 3a (page 34) and Table 3b (page 38). Since there are some surprises and discoveries that may not jump out at you from all those numbers, we want to discuss some of the more interesting (in some cases, bizarre) results. We'll discuss them by functional grouping.

In the area of screen I/O, none of the compilers was so fast as to indicate that the library routines were writing directly to screen memory (which could be a portability problem), but there was a wide range in performance. The compilers seemed to separate into two groups. Mark Williams' compiler was the fastest in a group clustered around 30 seconds, with MicroSoft's the fastest in a group clustered around 55 seconds. This suggests the use of two divergent methods of handling screen I/O under MSDOS.

Also, we were surprised to find that for most compilers, prtf( ) was not much slower than scroll( ). We had expected the conversion overhead to be significant, but this was true only with Computer Innovations' compiler and, to a lesser extent, with DeSmet's. A real anomaly was Digital Research's compiler, whose prtf( ) time was faster than scroll( ) and even faster then fillscr( )! That means that in DRI's compiler there is some overhead associated with using puts( ) that is avoided in using printf( ); this made no sense to us.

The storage class tests basically tested the efficiency of loading data from memory. Since there are only a limited number of ways to load a two-byte quantity from memory, it is not surprising that there were many-way ties in this category. In fact, except for the two slowest compilers (c-systems' and Software Toolworks'), all the compilers were closely ranked for all three tests. There were four compilers tied for first, followed by two more just a couple of seconds slower and five more tied a bit further off. Mark Williams' and Control C's compilers were the two tied in the second group, but that's no surprise; they tied in every test. They are really the same compiler.

If we compare the regtest( ) and autotst( ) results, it becomes clear that all the faster compilers implemented register variables, while generally the slower compilers did not. The c-systems compiler, the slowest for these tests, did implement register variables, but apparently to no great advantage. Out of curiosity, we also ran a version of regtest( ) that declared only two of the four loop variables to be of register class. For all compilers, the results were the same as for regtest( ). This was educational; it means that, at most, only the first two variables declared were actually kept in registers. This has implications for programming practice: use register declarations, but take

care that variables be declared with the most frequently used first.

For all compilers, the automatic storage class is more efficient than the static. This may come as a surprise to those accustomed to C on 8-bit machines with no stack-relative addressing mode.

Most compilers performed about the same on the looptst( )benchmark—about 60 seconds. The Aztec compiler, however, was significantly faster at 38 seconds. It is unusual that no one else came close to this time, since most benchmarks had several compilers within a few seconds of the fastest time. It turned out that the Aztec compiler generated only two instructions for the empty inner loop, whereas the best the others could do was three instructions.

The Aztec compiler repeated this performance on the array( ) test, coming in twice as fast as the nearest competitor. Here Aztec's performance was due to avoidance of the **imul** instruction in doing subscript arithmetic. In general, the Aztec compiler avoided **imul** in favor of shifts and adds wherever possible. The Aztec compiler also shared the lead in the pointer( ) test with MicroSoft's and Wizard's.

There were no surprises in the tint( ) test. Except for Software Toolworks' C, all the compilers were in a very narrow range, clustered around 35 seconds. There are not many ways to add or subtract two 16-bit integers on a 16-bit machine, and we did not use register declarations in tint( ), which would have given a slight edge to the compilers that support them.

Tlong( ) is another story altogether. The Datalight compiler had its only win here (by a significant margin) at 88 seconds, followed by MicroSoft's and then the Aztec compiler. The real standout in this benchmark was Digital Research's compiler, which came in last by an astounding margin with a benchmark of 808 seconds! It is hard to imagine how it could execute this slowly, over nine times as long as the fastest time. It is especially puzzling because the DRI compiler was a respectable performer otherwise, in fact fastest or essentially tied for fastest on a number of the other tests. If you use longs very much, DRI's is clearly not the compiler of choice!

The results of the optimize( ) benchmark are difficult to interpret. Some of the "lightweight" compilers ran this test quite well, while MicroSoft's did rather poorly, even though we had observed that its optimization was generally very good. Also, the Lattice compiler was fastest by a large margin on this test but was not an outstanding performer on any of the other benchmarks. We looked at the code generated by some of the compilers for a clue and discovered a partial answer. Optimize( ) was especially unkind to those compilers that do not recognize that multiply operations can be replaced by left shift instructions. One saved **imul** instruction can make up for a lot of redundant load and store operations! Actually, this kind of test should always be approached with caution. One of these days a compiler may come along that is so good at global optimizations that it notices when our benchmark functions produce no side effects visible outside the function. It might then be free to replace an entire function with a single RET—very fast indeed!

Turbo Pascal Programmers

The most comprehensive **screen management and display** system *on the market,* bar none, now speaks Turbo.

Full keyboard editing features
Mono and graphics display support
Time/date (dynamic) display option
128 fields per screen, 30 screens per file
40 function and 20 interrupt keys per screen
NumLock, CapsLock, and Scroll status display
Validity check, multiple ranges & values per field

S3-TScreen is a friendly post-processor for IBM's *Application Display Management System.* (IBM's PC-DOS languages, BASIC, PASCAL, COBOL, FORTRAN, and Assembler, already interface with *ADMS.*)

S3-TScreen generates *ADMS* linkage for Turbo by writing two sources files, an INCLUDE file and a (very small) main program, which can be immediately compiled and executed. One FUNCTION call clears or displays the screen of your choosing. Screen input is accessible to your Pascal code by individual variable, via field names assigned in *ADMS,* and by record. Just add your application code to the main program.

S3-TScreen is **$29.95.** Master Card is accepted.

Social & Scientific Systems, Inc.
Attention: **S3-TScreen**
7101 Wisconsin Avenue, Suite 610
Bethesda, Md. 20814
301-986-4870

Circle **no. 124** on reader service card.

Introducing SMK, the

SEIDL MAKE UTILITY™

When we at S.C.E. needed an MS-DOS make utility for in-house use, we couldn't find one that did everything we needed... so we wrote SMK. Now we are offering SMK at a fantastic introductory price!

Advanced SMK features include:

- Proprietary dependency analysis algorithm analyzes all dependencies before rebuilding any files.
- SMK understands complicated dependencies involving nested include files and source and object code libraries.
- High-level dependency definition language makes setting up dependencies easy. Supports parameterized macros, local variables, constants, include files, command line parameters, line and block comments.
- Batch source code editor that allows automatic source file updating is included.
- FAST! SMK can analyze hundreds of dependendies in just seconds.
- Typeset user's manual and excellent error diagnostics make SMK easy to learn and easy to use.

Why waste valuable product development/maintanance time doing work that SMK can do for you? Order SMK today!

SMK Introductory price, save 40%: $84.00

SMK List price (effective Jan 86): $140.00
(include $3.50 postage & handling)

Multi-site License
Dealer Inquires
Educational Discounts

Seidl Computer Engineering
1163 E. Ogden Ave., Suite 705-171
Naperville, IL 60540 (312) 983-5477

Circle **no. 114** on reader service card.

The disk I/O benchmarks were really testing part of the library, not the compiler, so we expected a wide range of numbers. What we did not expect was that for many of the compilers, cpychr( ) would be faster than cpyblk( ). When you look at Table 3b, don't overlook the loop count: cpyblk( ) moved twice as many bytes as cpychr( ). The MicroSoft compiler had the fastest time for cpyblk( ), with no other compiler coming close. The Aztec compiler was fastest for cpychr( ), with Computer Innovations' the only close second. The Software Toolworks compiler was abysmally slow on both of these, about four times slower than the generally poor-performing compiler from c-systems. On the other hand, the Toolworks compiler gave a reasonable time for diskio( ), while the fastest time, by a wide margin, was turned in by c-systems' compiler. In fact, the compilers that generally did the best on the various compile-time benchmarks (those from Aztec, MicroSoft, and Wizard) were among the slowest on the random-access diskio( ) test. You figure it out!

The min–( ) benchmarks were intended to test the granularity of the library: we wanted to see how much unwanted "stuff" would be dragged into the .EXE file along with the library functions actually being called. Minmain( ) contains no function calls at all, so the .EXE file size for this test is an indication of the best you can do. The DRI, c-systems, and Lattice compilers were the big losers here, all requiring more than 10,000 bytes of library code to do nothing at all. Not surprisingly, these three were also the worst on the minputs( ), minprtf( ), and minfio( ) tests, although the Lattice compiler's .EXE size remained about the same. The EcoSoft and Wizard compilers both produced amazingly small .EXE files for minmain( ), since the startup code should at least initialize the stdin, stdout, and stderr files. DeSmet's, in third place at 1536 bytes, already includes all the library code needed to support the puts( ) function. That made DeSmet's the smallest .EXE file for minputs( ) by a fair margin, although Wizard's and EcoSoft's were also very small.

Minprtf( ) was the acid test, because the formatting supported by printf( ) usually requires the floating-point package as well as I/O support. Some of the compilers provide an integer-only version of printf( ), however, and we used that option whenever possible since we believe the majority of C applications do not require floating-point support. The compilers that provide this option are giving you more flexibility with an effectively more granular library. They are the compilers from Aztec, Mark Williams (and Control C), Lattice, MicroSoft, and Datalight.

The MicroSoft compiler deserves special mention among these. The other compilers made you specify whether floating-point support was desired, either with a command-line option or via library-search order. MicroSoft's handled this automatically by detecting whether any float or double variables or pointers had been declared in your program, and then linking the math library only if needed. We found this sort of feature to be typical of the Microsoft product.

Using the floating-point version of printf( ) typically added from 2000 to 3000 bytes to the .EXE file size. Choosing the compile option to leave the float library out, Aztec C produced the smallest .EXE file by a large margin, even over the integer-only competitors. Datalight's and MicroSoft's came next and then DeSmet's. The DeSmet compiler produced the smallest .EXE file with the floating-point library included, although the Aztec file was only a few hundred bytes larger. Also, it should be noted that the Software Toolworks compiler is a subset compiler with no floating-point support at all.

For the minfio( ) benchmark, the Datalight, Aztec, and EcoSoft compilers shone the brightest, with Wizard's not far behind. Here DeSmet's .EXE file was a lot larger than the one generated for minprtf( ), even though the float library was not required. This was viewed by all as very strange.

The min–( ) benchmarks, except for minmain( ), are all significant because they give you an idea of how small a small program might be. Since there are very few useful functions to be performed that do not require some console and disk I/O, minmain( ) was included primarily to show the losers in this category in the unkindest light possible. The c-systems and DRI compilers did not fare well at all, requiring 19K and 18K, respectively to say "hello world" via printf( ). Lattice's compiler, a very well thought of product generally, also looked none too good here. The Aztec compiler was the overall winner in this category. The Aztec compiler also had the smallest execution-time benchmark files, requiring only 14K for benchall.exe versus an average of 55K–65K for the others.

Memory Models

The 8086's segmented architecture leads to some tradeoffs between execution speed and addressing range. The way a compiler views the 8086's address space is called the memory model. All the compilers reviewed here support the so-called "small" memory model as the default, and many of them allow no other. The small model allows up to 64K for instructions and 64K total for all kinds of data: heap, stack, statics, and globals. Automatic variables are allocated on the stack, with space dynamically allocated via malloc( ), etc., coming from the heap.

Other possible models are called different things by different manufacturers. Lattice, Aztec, c-systems, Wizard, and Microsoft all provide a model in which code (instructions) can run up to a megabyte (the maximum allowed by the 8086), but data is limited as in the small model. Lattice, Aztec, c-systems, and Wizard provide a model that limits code as in the small model, but the heap may extend to a megabyte, the stack can grow to 64K, and statics and globals together can take up 64K. DRI and Computer Innovations, in addition to the just-mentioned four, provide a model allowing code up to a megabyte and data as in the preceding model. Wizard and Microsoft have a model that extends this last model by allowing more than 64K of combined statics and globals (although no single item may be greater than 64K), in addition to the one-megabyte code and heap spaces; the stack is still limited to 64K.

Floating-point and 8087 Results

Vendor	tflt	tdbl	EXE size	8087	inline	Notes
8087 series:						
Aztec (Manx)	119.1	141.0	7584	Y	N	m87.lib
Aztec (Manx)	149.1	168.5	10000	N	N	m87s.lib
c-systems	145.0	153.7	22424	Y	N	
c-systems	59.9	68.3	20956	N	Y	
Computer Innovations	59.2	68.3	10027	N	Y	
DeSmet	119.0	128.3	12288	N	N	stdio7.s
Digital Research	67.3	76.5	22528	N	Y	
Ecosoft	290.5	161.7	10828	Y	N	
Lattice	1675.8	1256.0	17452	Y	N	didn't use 8087
Mark Williams	53.7	62.9	9632	N	Y?	-rndp -vsmall
Microsoft	81.0	89.7	21172	Y	N	/fpc
Microsoft	81.0	89.6	14148	N	N	/fpc87
Microsoft	55.7	64.2	20932	Y	Y	/fpi
Microsoft	55.6	64.2	13908	N	Y	/fpi87
Wizard	350.0	209.0	11788	Y	N	
Wizard	368.7	72.0	11916	N	Y	-f
Floating-point series:						
Aztec (Manx)	78.8	81.0	9136	N		m87.lib
Aztec (Manx)	80.6	83.2	10000	Y		m87s.lib
c-systems	95.1	78.6	22424	Y		
Computer Innovations	173.1	166.0	11453	N		
Datalight	121.8	71.1	10068	N		
DeSmet	97.8	79.2	14848	N		
Digital Research	83.6	61.7	22016	N		
EcoSoft	129.6	105.8	10878	Y		
Lattice	112.0	84.2	17452	Y?		
Mark Williams	98.8	94.3	13137	N		
Microsoft	91.7	101.9	21172	Y		/fpc
Microsoft	25.6	45.6	16580	N		/fpa
Microsoft	92.6	102.6	20932	Y		/fpi
Wizard	23.1	71.4	11788	Y		

The floating-point series are with loop counts of 600, run at 7.16MHz.
The 8087 series are with loop counts of 6000, run at 4.77MHz.
The rightmost column indicates whether in-line code is supported. The column headed "8087" indicates, for floating-point results, whether the test runs faster with an 8087 present, and, for the 8087 results, whether it works if an 8087 is not present.

Table 4
Floating-point and 8087 Results

Documentation and Support:

Vendor	Usability	Readability	Support
Aztec (Manx)	2.83	**3.72**	1.88
Control C	3.17	2.87	1.56
c-systems	2.50	2.55	2.19
Computer Innovations	3.17	2.83	2.50
Datalight	1.00	1.03	.63
DeSmet	2.33	2.17	3.44
Digital Research	3.33	2.58	**4.06**
EcoSoft	2.17	2.05	1.56
Lattice	3.67	2.97	2.81
Mark Williams	3.17	2.87	1.88
Microsoft	**4.67**	**4.02**	3.13
Software Toolworks	1.83	1.33	1.56
Wizard	3.33	2.85	**4.38**
Interpreters:			
C-terp	2.50	1.83	.63
Instant C	3.00	1.25	1.56
Run/C	2.50	**3.87**	.94

Table 5
Documentation and Support

TURBO Power

PROGRAMMER'S UTILITIES
especially for Turbo Pascal on IBM PC/XT/AT and compatibles

MORE POWERFUL THAN UNIX UTILITIES!!!

These Ready-to-Use programs fully support Turbo Pascal versions 2.0 and 3.0, and PCDOS 2.X and 3.0. Here's what you get:

Pretty Printer
Standardize capitalization, indentation, and spacing of source code. Don't waste your own time! Several adjustable parameters to suit your tastes (works with any standard Pascal source).

Program Structure Analyzer
Find subtle problems the compiler doesn't: uninitialized and unused variables, modified value parameters, "sneaky" variable modification, redefined standard identifiers. Also generates a complete variable cross reference and an execution hierarchy diagram. Interactive or write to file (works with any standard Pascal source).

Execution Timer
Obtain a summary of time spent in each procedure and function of your program, accurate to within 200 microseconds. Also counts number of calls to each subprogram. Fully automatic.

Execution Profiler
Obtain a graphic profile of where your program spends its time. Interactive, easy-to-use. Identify weak code at the instruction level. (Profiler and Timer for Turbo Pascal Source code only.)

Command Repeater
Customize any operation by reading and parsing the standard input. Send up to 255 keystrokes to any executed program. Automatically generate DOS batch files.

Pattern Replacer
Find and REPLACE versatile regular expression patterns in any text file. Supports generalized wildcards, nesting, alternation, tagged words and more. Over a dozen programmer's applications included.

Difference Finder
Find differences between two text files, and optionally create an EDLIN script which rebuilds one from the other. Disregard white space, case, arbitrary characters and Pascal comments if desired.

Super Directory
Replace PCDOS DIR command with extended pattern matching, sort capability, hidden file display, date filtering, and more.

File Finder
Locate files anywhere in the subdirectory tree and access them with a single keystroke. Display the subdirectory tree graphically.

AVAILABLE IN SOURCE AND EXECUTABLE FORMAT

Executable: $55 COMPLETE including tax and shipping. Compiled and ready to run, includes 140-page printed user manual, reference card and one 5¼" DSDD disk. Ideal for programmers not using Turbo. NOT copy protected.

Source: $95 COMPLETE including tax and shipping. Includes all of the above, and two additional DSDD disks. Disks include complete Turbo Pascal source code, detailed programmer's manual (on disk) and several bonus utilities. Requires Turbo Pascal 2.0 or 3.0.

Requirements: PCDOS 2.X or 3.0. 192K RAM — programs run in less RAM with reduced capacity. Two drives or hard disk recommended.

TO ORDER:
VISA/MasterCard orders, call 7 days toll-free 1-800-538-8157 x830. In California, call 1-800-672-3470 x830 any day.
Or mail check/money order to:

TurboPower Software
478 W. Hamilton Ave., Suite 196
Campbell, CA 95008

For technical questions, call 408-378-3672

Circle **no. 81** on reader service card.

Epsilon

The Emacs-Like Text Editor For Programmers Who Don't Like to Wait!!

State of the Art Text Editor

Epsilon is an exciting new text editor designed to make programmers more productive. Epsilon is faster than Brief, faster than Mince, faster than Gosling Emacs, *and faster than the editor you're using now!*

Concurrent Processes!

Epsilon lets you compile while you edit! You can run compilers, assemblers, linkers, and almost any other program that isn't screen oriented, all under Epsilon's control, while you edit your files!

With Epsilon you don't wait for programs like compilers to finish. Use Epsilon's concurrent process command, and while the compiler runs, you can continue to examine and edit files. Any errors in the compilation are displayed immediately, and Epsilon gives you the opportunity to correct them **while the compiler continues to run.** *With Epsilon, you're finished correcting errors when other editors first let you start.*

Powerful Commands

Epsilon has over 125 commands instantly available. Epsilon can manipulate words, sentences, and paragraphs easily. Epsilon will automatically save text you have deleted in a "ring" of kill-buffers, so that you can retrieve it later. It will help you avoid syntax errors by displaying matching parentheses. *And best of all, Epsilon's macros let you define your own commands, which can be loaded automatically each time you start Epsilon.*

Speed with No Limits.

Epsilon reads and writes files 25% to 600% faster than competing editors. From its convenient keyboard macros to its facility that completes the names of commands, files and buffers, to its optimized incremental search, Epsilon has been designed for programming ease and speed.

There's no limit to the number or the size of buffers you can have. Each buffer can hold a different file, or different versions of the same file. You can create as many windows as will fit on the screen, and display different buffers in each. And should you run out of memory, Epsilon will create and automatically utilize a swap file.

Speed Comparison (In Seconds) with Other Editors

	Epsilon	Brief	Mince	Emacs
Start-up	2.60	4.11	1.43	24.93
Read 21K file	1.06	1.33	8.95	7.52
Write 21K file	2.11	14.30	6.05	7.95
Next Screen	.19	.24	1.33	1.80
String Search	3.85	7.04	4.49	8.41
I-Search	3.85	--	--	8.73
First Help	8.30	12.33	--	--
Other Helps	.20	11.64	--	--

Epsilon runs on IBM PC's, XT's, AT's and compatibles with PC-DOS 2.0 or above and requires 192K of memory.

Epsilon's price is only $195.00.

ALL MAJOR CREDIT CARDS ACCEPTED.

Lugaru Software, Ltd.

5227 Fifth Avenue, Suite 12 / P. O. Box 110037
Pittsburgh, Pa. 15232

(412) 621-5911

Circle **no. 6** on reader service card.

The Wizard and Lattice compilers each have an option that permits the compiler to generate much better code for the large models, if the programmer is willing to live with certain restrictions regarding pointer arithmetic. These two compilers also allow very small programs to be changed to .COM files. Such programs must fit all code and data into 64K.

Microsoft is currently the only one of the 13 reviewed manufacturers supporting "mixed-model" programming, although others indicate that it is high on the to-do list. Mixed-model programming allows certain pointers to be declared "near" or "far," allowing you to use a pointer from a model different from the model of the rest of the program. You can write a program that uses the small model with its speed benefits and make only certain pointers of the time-consuming "far" variety to gain access to very large arrays.

Floating-Point and 8087 Support

Although C is presently most often applied to problems that don't require any floating-point arithmetic, the language does support and define the use of "real" numbers. Of the 13 compilers reviewed (see Table 4, page 43), all but one, Software Toolworks C, support floating-point math, and Toolworks expects to have it available by the time you read this.

The remaining 12 all support both float and double data types as 32- and 64-bit quantities, respectively. Nearly all use the IEEE floating-point standard to represent float and double. This makes it relatively easy to take advantage of an 8087 math chip (which also uses the IEEE standard). Datalight's is the only compiler we reviewed that did not offer any way to take advantage of the 8087. Lattice's compiler documentation indicated that its library was "sensing," but our results indicated that it did not use the available 8087.

A sensing library is a library of floating-point functions that use the 8087, if one is present, and do the required operations in software if not. This means that you, the programmer, don't have to know in advance if your program is going to run on a machine with an 8087.

Another way of supporting an 8087 is to use a library that requires the presence of an 8087. Compared to the sensing library, this buys you a little time (the time the sensing library takes to decide whether there's an 8087 for it to use) and some 2K–7K of code (the software floating-point routines). Or you can generate 8087 instructions in-line like integer operations. This saves the time needed to make the function call.

It is possible to execute an 8087 instruction in a machine not having an 8087. What happens is that the 8086 traps to a user-provided interrupt routine. This routine may emulate the 8087 and return. This allows the compiler to generate in-line instructions and still not require an 8087, but to use one if it's available. Only Microsoft's compiler implements this type of an emulation scheme.

Documentation

We reviewed all documentation and packaging for each

compiler. To be consistent, we developed "benchmarks" for documentation, with a grading system consisting of two main categories: usability and readability.

Usability

The usability category covered the overall packaging of the documentation: the organization and quality of presentation that make the documentation easy or difficult to use. The items that contributed to the usability score were the following:

- Packing list (P/L): Was a list of all manuals, disks, etc., included so that you could confirm that you had received everything?
- Disk contents (DISK): Was a list of all disks and their contents included? Often one was included, but buried in the manual and hard to find.
- Binding (BIND): How was the documentation received from the vendor? We saw everything from loose pages (not even clipped or stapled) to multiple 3-ring binders in attractive cases.
- Table of contents (TOC): Was there one? How complete was it?
- Index (INDEX): Was an index included, and how complete was it? Believe it or not, some vendors don't want you to locate anything in the documentation, as proven by the fact that they don't index their manuals!
- Error codes (ERR): Was a list of error codes generated by the compiler present, and how useful was it?
- Startup (START): Was a getting-started section supplied for those who need this level of support? Also considered here: the quality of instructions on batch files for installation, sample "test" programs to try out the compiler, and sample batch files for operation.
- Operation (OPER): Was a compiler-operation section included, how complete was it, and could you find each compiler option without rereading the entire section?
- Libraries (LIB): Was a section describing the libraries supplied, and how easy was it to find a particular function? This is where a good index helps, but at least the functions ought to be in some order, not at random! The CI-86 compiler included a brief summary of the functions in its table of contents—bravo!

Readability

Readability pertained to clarity of presentation and to the ease with which the material could be understood. To establish some method of consistent grading, we picked three library functions and subjectively graded the presentation on each function across all compilers. We picked fopen( ), fread( ) (where supported), and printf( ). We looked for such things as good examples, clear definition of all parameters, clear definition of return values, error returns, notice of deviation from standards, and a good explanation of the function. We also gave credit if only one function was printed on a page, making it easier

Time and Money.

We've just done something we know you'll like. We've made the SemiDisk far more affordable than ever before. With price cuts over 25% for most of our product line. Even our new 2 megabyte units are included.

It's Expandable

SemiDisk Systems builds fast disk emulators for more microcomputers than anyone else. S-100, IBM-PC, Epson QX-10, TRS-80 Models II, 12, and 16. You can start with as little as 512K bytes, and later upgrade to 2 megabytes per board...at your own pace, as your needs expand. Up to 8 megabytes per computer, using only four bus slots, max! Software drivers are available for CP/M 80, MS-DOS, ZDOS, TurboDOS, VALDOCS 2, and Cromix. SemiDisk turns good computers into **great** computers.

Battery Backup, Too

At 0.7 amps per 2 megabytes, SemiDisk consumes far less power than the competition. And you don't have to worry if the lights go out. The battery backup option gives you 5-10 hours of data protection during a blackout. Nobody else has this important feature. Why risk valuable data?

The Best News

	512K	1Mbyte	2Mbyte
SemiDisk I, S-100	$695	$1395	
SemiDisk II, S-100	$995		$1995
IBM PC, XT, AT	$595		$1795
QX-10	$595		$1795
TRS-80 II, 12, 16	$695		$1795
Battery Backup Unit	$150	$150	$150

Someday you'll get a SemiDisk. Until then, you'll just have to....wait.

SEMIDISK

SemiDisk Systems, Inc., P.O. Box GG, Beaverton, Oregon 97075 503-642-3100

MasterCard VISA

Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards (300/1200 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle **no. 85** on reader service card.

to find and read. Cop-outs such as "see K&R for description of this function" sat very poorly with us.

Table 5 (page 43) shows the numerical scores we arrived at for documentation. The Microsoft manuals were the best on virtually every point. The Wizard, Lattice, and DRI manuals should be usable over the long run, and the Aztec book was very readable.

User Support

The best software product is of marginal value if it is not well supported. We attempted to review the kind and quality of user support available for the compilers reviewed. This included contacting support groups for most of the compiler publishers to evaluate their responsiveness.

Although you should not expect the same level of support for an MSDOS compiler as you might for a compiler running on a large mainframe, some of the same support features should be available. Ideally, support should reflect some modularity, so those who need full support can obtain it, while those who need only occasional assistance don't have to pay for support they won't use. We attempted to evaluate user support based upon several criteria that we determined to be important. The numerical results are shown in Table 5. Here is a discussion of the criteria.

Bug List

Most of the compilers did not have available a *current* list of known bugs. Some did indicate problems known but not yet fixed as of a particular release, either in a read.me file or in the documentation. Ideally, what we wanted to see was a regular update describing current bugs, the expected fix date, and any work-arounds. This would be a big help at two in the morning when you are under pressure to complete a big project, but are stopped dead by a compiler bug. It also ought to help the customer service group by reducing the number of times the same bug is reported.

Registration

Most vendors supplied a means to register the compiler so that the buyer could receive notices of future releases. Some were even postpaid. We had no means of verifying that new release notices would appear in the mail, but past experience indicates that sometimes such notices are sent and sometimes not, depending on the vendor.

Update Subscription

Only one vendor (Wizard) offers a fixed-price annual subscription service for all bug fixes and updates. Many others indicated that updates were available but were charged on a per-update basis. The Wizard approach has

NETWO

SOMEBODY HAS BEEN DOING SOMETHING TO ANSWER YOUR

Once again VersaSoft is answering your dBASE needs with the FIRST WORKING dBASE compatible multi-user database manager for local area networking.

When Ashton-Tate introduced Multi-User dBASE II over a year ago, it crashed in flames. Since then none of the dBASE-compatible vendors have even talked about networking. If you're waiting for dBASE III, dB/Compiler, cEnglish, FoxBase II or Clipper to answer your networking needs, you may be in for a long wait. Now, VersaSoft has succeeded where others have failed, or haven't even started... with dBMAN-Net™. All the advanced features of dBMAN-Net™ make the elusive promise of networking a reality, today! You may have been tantalized by a product announcement before, only to find out it will be months before the product is on the market. By contrast, dBMAN-Net™ has already been field proven, in fact, several commercial on-line database systems have been using it reliably for months. The performance has proven to be excellent and dBMAN-Net™ is fully compatible with the single-user dBMAN™.

dBMAN-Net™ provides all the tools you need to attack any networking database applications: Record and file locking; semaphore locking; locking many records from one file and/or from different files as a "set". (All the records you specify are either locked or nothing is locked, avoiding deadlock.) dBMAN-Net™ also locks, unlocks, and updates the index files automatically, all you do is lock the database and dBMAN-Net™ does the rest! Existing dBASE programs are easily converted to multi-user with two simple new commands, DENYRW and UNDENY. Simply use DENYRW to lock the record before the REPLACE command and use UNDENY to unlock the record. dBMAN-Net™ also provides dBASE programs full control over error trapping and recovery. A printer output can be queued and spooled to the file server and different print jobs can be printed in an orderly manner on any of three shared printers. dBMAN-Net™ is the only networking database with full print server support built in so you control everything from banner text, copies, tab expansion and printer to form selection.

dBMAN-Net™ runs on Novell NetWare v4.6x Local Area Networks, the IBM PC, XT, Portable, AT, and is compatible with 256K RAM. NetWare support is available for most LAN hardware including Gateway, IBM, 3Com, Orchid, and many others.

dBMAN-Net™ comes with a site license with NO ADD-ON CHARGE, allowing more users as your network grows. Run-time module is now available and the site license for it is just $150.00 per network.

VersaSoft provides you with a full line of database products to fill your application requirements, including single user dBMAN™, dBMAN™

appeal: it's simple, and you should receive updates faster, with less paperwork at both ends.

Phone Consultation

Phone-in consulting service was available from all vendors, but some vendors required a serial number to gain access and others required money before giving the buyer a phone number to call. Although it is certainly not unreasonable to charge for support, many vendors bundle phone support into the basic product. We found that some vendors restricted calling hours.

Support Quality

All the reviewers made notes when they called vendors for support, and the documentation group also attempted to contact the support personnel for each vendor. We found a wide range in quality of support. Sometimes we were referred to the author of a compiler. That should get you qualified technical support, but we felt it to be a negative; it meant the technical support was dependent on one person, and if that person is unavailable, you're stuck. On the other hand, talking to a "support person" who doesn't know C makes one very uneasy. We found that the support team at Wizard offered by far the best technical assistance.

In Table 5 you see an attempt to assign numbers to our experiences with the vendors in these areas of technical support. The clear winners here were Wizard's, DRI's, DeSmet's, and Microsoft's compiler support.

Compiler-specific Comments

The following comments, arranged by compiler, elaborate on the major items not already discussed or easily discernible from the tables: the quality of the documentation, standards issues, user support, overall benchmark performance, and summary remarks.

Aztec C86

Aztec C86 was easily one of the fastest compilers overall and handled array referencing more efficiently than any other compiler. Our results indicated that its library provides a lot of flexibility, allowing you to generate small .EXE files. It was the best in this area. The Aztec compiler provides an integer-only version of printf( ). There are several large-memory models to choose from. A make utility is included.

There was a general problem with the benchmarks: the main( ) routines used a cast to produce a null pointer to function returning an int — (int (*)( )) 0. Some compilers could handle it; many others couldn't. Aztec C86 couldn't handle this function cast in main( ), but it gave an excellent error message when it tried.

The Aztec manual had one major problem: it has no

RKING

dBASE NEEDS. VERSASOFT IS PROUD TO ANNOUNCE dBMAN-NET.™

run-time, multi-user dBMAN-Net™ and dBMAN-Net™ run-time. Furthermore, Versasoft products are NOT copy protected and all products are available **NOW.**

dBMAN-Net™ v1.04
Interpreter (Unlimited Users) — $1,100.00
Run-time Developer Package (Interpreter, Code generator and Site License) — $1,300.00
Run-time Code Generator (Interpreter is a prerequisite) — $200.00
Run-time License (Unlimited Users) — $150.00

dBMAN™ v1.04
Interpreter — $292.00
Run-time Code Generator (Interpreter is a Prerequisite) — $150.00
Run-time License (optional) — $20.00

dBase, dBASE II and dBASE III are trademarks of Ashton-Tate cENGLISH is a Trademark of cLINE
dB/Compiler is a Trademark of WordTech Systems Clipper is a Trademark of Nantucket
FoxBase II is a Trademark of Fox Software

dBMAN-NET™

YOU'VE BEEN IN THE DARK LONG ENOUGH.

VERSA OFT™ VersaSoft Corporation

723 Seawood Way, San Jose, California 95120 **(408) 268-6033**
Telex 6502635806 (Via WUI) Answer back MCI

Circle **no. 120** on reader service card.

index. Otherwise it rated above average, though we thought it needed more white space for readability. Aztec did include a section on making your code ROMable, very useful to some, and it had good examples in the library descriptions, although these tended to describe more than one function at a time. This does save repeating the same text over again, but it makes it a little difficult to extract specifics about a particular function. A new manual is available (received too late for review); among the utilities included are editor, debugger, make, Aztec to .OBJ format converter, CRC utility, archiver, squeezer, grep, and hex generator.

Control C CC-86

Control C's compiler is a copy of Mark Williams C (MWC), documentation and all, so we'll refer you to our comments on MWC. Apparently, Control C is doing the port of the Mark Williams compiler to CP/M-86 and, as part of the arrangement, is able to provide the MSDOS version to the outside world. The only comments we'll make specific to the Control C compiler are that one of the six disks in the original package we received had a defect and that the Control C package did not include the function-key pad overlay for the debugger, as the Mark Williams package did.

c-systems 8088 Software Development Package

Although generally slow, the c-systems compiler turned in the fastest time in the disk I/O test. It did not do well in tests of generated-code size. c-systems provides several large-memory models. The c-systems compiler does not support fgetc( ) in stdio.h, even though fputc( ) is supported. Consequently, fgetc( ) was coded as a module, rather than as a macro, using getc( ).

The c-systems manual contained a neat trick for compiling multiple modules using wildcards. Other compilers support wildcards in the command line; c-systems uses the **for** batch command from the command line. For example:

```
A>for %f in (min*.c) do cc %f
```

This technique can be used with any compiler. c-systems' compiler was about average in the area of documentation. It didn't include a binder, and the print quality was only fair. The installation section was too brief, but the operation section was well done. It did include a detailed list of differences from the standard, something many did not. The library descriptions lacked examples and could use more information on return and error values. There were problems using drivers; occasionally they would not run for no apparent reason and they only used the first 128 bytes of the environment string so the INCLUDE variable was not always findable; the compiler has PL/M compatibility.

Computer Innovations Optimizing CI-86

The CI-86 compiler performed in the middle range on nearly all the benchmarks, walking away with the copychr disk I/O test. One nice feature of the CI-86 compiler is that numerous paths are provided for interfacing to the operating system. For instance, you can initialize and release interrupt routines such as might be required for writing a communications program.

Computer Innovations provides a large-memory model, although when we used it, it would not allow pointer arithmetic on a multiply-dimensioned array across dimensions. The documentation explains this strange behavior, stating that without the functions that convert pointer to long (ptrtoabs( )) and long to pointer (abstoptr( )), "big-model pointer arithmetic assumes the data segments are the same for the pointers." The compiler did give a reasonable error message when an input data file was not found. The CI-86 compiler doesn't allow ++ or -- on float or double variables. For example:

```
double d; d++;
```

did not work, while

```
double d; d += 1;
```

did work.

We judged the Computer Innovations manuals above average. They included a good approach in the table of contents: a one-line description of each library function, serving as a quick-reference guide. Their library descriptions were well done, with examples and notes on DOS applicability. The compiler-operation section was too brief, and no listing of compiler error messages was included.

Datalight C

The old MSDOS Small-C compiler distributed by Datalight was a port of the public domain Small-C compiler for CP/M. The new C Compiler, which is a full implementation of the C language, stood out in a number of areas. It came in first or tied for first on several tests, standing in a class by itself on the long-arithmetic benchmark. Datalight provides an integer-only version of printf( ).

The Datalight compiler appears to implement real block scope for declarations. Function xmalloc( ) was declared as extern char *xmalloc( ) inside main, and this declaration was not in force in later functions, leading to a "definition inconsistent with prior use" error when the function was called. We view this as appropriate, but it is not clear what behavior K&R calls for, and most of the other compilers did not catch this error.

The Datalight compiler could not compile the global definition

```
char null_strg[ ] = { "" };
```

It had no problem when it was changed to

```
char null_strg[1] = { "" };
```

The documentation that came with the new Datalight compiler was a significant improvement over the previous version, as was the compiler itself. The packaging was still

low-budget, like the compiler—56 8½-by-11 spiral-bound pages with small, difficult-to-read print. All the essentials were there, though. There was a usable table of contents and an index, the few supported error messages were all described, there was a setup section with a sample program, and some of the function definitions included examples. Overall, not bad for a $50 compiler.

DeSmet C

The DeSmet compiler was generally mid-range in speed, coming in above average in the screen I/O tests and losing the floating-point math test. It produced the smallest .EXE files in certain of the code-size tests. The compiler is supported by a good user group, and there was a bug list.

No error message was produced when an input data file was not found—the compiler just went to sleep! DeSmet's compiler does not like really long comments; an error message was generated at line 28 after the opening /*.

DeSmet's manual is a low-budget product. Its pages were not bound nor even stapled; they lacked an installation section, and although the DeSmet package supplied a lot of software, nowhere is there a list of what you should have received in the package. The manual also omitted an index and didn't organize the functions as we would have liked, with one function on a page. The descriptions were brief, but they did include examples.

Digital Research DRI C

Although the DRI compiler was among the faster overall, it dawdled through long arithmetic. The compiler performed poorly on the tests of library granularity. DRI provides a large-memory model, and DRI's support got high marks.

DRI's compiler had no problem with the function cast in main( ). No timer function and no access to DOS interrupts were provided; an assembler is required, but rasm86 was included, and the manual documented the assembly and linkage process very well, including an example. Capital letters on the command line were converted to lower case before being passed to main( ).

DRI's compiler was the only one to report an error for two statements in doc1000.c that defined and initialized global arrays. Both initialization lists included a trailing comma, as in:

```
int array[ ] = { 1, 2, 3, 4,
        5, 6, 7, 8,
    };
```

The output file produced by cpyblk( ) was too small because fread( ) returned zero instead of the actual number of bytes read on the final block. The block size was 1024 bytes, so the last block would have been a short one.

Digital Research put a good documentation package together. They provided a reprint, in their own binding, of K&R. The index was good, and a nice error list appeared in an appendix. The getting-started and operation sections were well done, with good tables and examples. The library function descriptions could have been better organized,

TURBO EDITASM

Introducing the first co-resident editor assembler for the IBM PC family. ***TURBO EDITASM (TASM)*** is significantly faster and easier to use than the IBM Macro-Assembler (MASM). Whether you are new to assembly language and want to quickly write a small assembly language routine, or are an experienced MASM user tired of waiting months to assemble large files, ***TURBO EDITASM*** will bring the excitement back to assembly language.

***TURBO EDITASM* IS MUCH FASTER:**

- How fast is ***TASM?*** The graph below shows relative assembly times for a 48K source file. For large files like this we blow MASM's doors off at 3 times their speed. For smaller 8K files we positively vaporize them at 6 times their speed.

  TASM (110 sec.)
  MASM (340 sec.)

- ***TURBO EDITASM*** is faster for the following reasons: **(1)** Written entirely in assembly language (unlike MASM). **(2)** Editor, assembler and source file always in memory so you can go instantly from editing to assembling and back. **(3)** Eliminates the time needed to LINK programs. Executable COM files can be created directly. (Also creates OBJ files compatible with the IBM linker).

***TURBO EDITASM* IS EASIER TO USE:**

TASM includes many other features to make your programming simpler.

- Listings are sent directly to screen or printer. Assemblies can be single stepped and examined without having to leave the editor.
- Access the built-in cross reference utility from the editor.
- Full support of 186 and 286 (real mode) instructions.
- Both Microsoft and 8087 floating point formats are supported. 8087 and 287 instructions supported directly without macros for faster assembly.
- Calculator mode: Do math in any radix even using symbols from the symbol table.
- Direct to memory assembly feature lets you test execute your code from editor.
- Coming soon: A coordinated symbolic debugger.

COMPATIBILITY: ***TASM*** is source code compatible with MASM and supports macros, records and structures.

Introductory Price $49
With .OBJ Capability $99

Speedware™

Include $5.00 shipping and handling. California residents add 6% Sales Tax.

Dealer inquires welcome

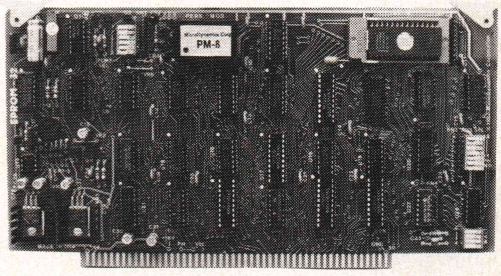
916-988-7426
118 Buck Circle, Box D
Folsom, CA 95630

IBM, Microsoft trademarks of IBM Corp., Microsoft Corp.

Circle **no. 59** on reader service card.

S-100 EPROM PROGRAMMER

EPROM-32



- Field-proven board meets IEEE-696 standard.
- Programs 1K through 32K (byte) EPROMs.
- Textool zero-insertion-force programming socket.
- EPROM is programmed through I/O ports and can be verified through I/O ports *or* located in memory space for verification.
- Programming voltage generated on-board.
- Personality Modules adapt board to EPROMs:

PM-1—2508, 2758	PM-3—2732, 2732A	PM-6—68764
2516, 2716	PM-4—2564	PM-8—27128
PM-2—2532	PM-5—2764	PM-9—27256

- Feature-packed CP/M-compatible control software includes **fast programming algorithm.**
- One year warranty.

$269.95*
(A & T)

MicroDynamics
Corporation

Suite 245 • 1355 Lynnfield Road • Memphis, TN 38119
(901)-682-4054

* Price includes EPROM-32, documentation and two Personality Modules(specify). Additional Modules—$7.95. Control software on 8" SSSD diskette—$29.95, UPS ground—$2.00, UPS air—$4.00, COD—$1.65, foreign add $15.00, VISA & MASTERCARD welcome.

See Dec. 1983 **Microsystems** for a review of the EPROM-32.

Circle **no. 73** on reader service card.

though, and should have included examples instead of directing the reader to K&R for descriptions of functions.

EcoSoft-C

The EcoSoft compiler was not dazzlingly fast nor distressingly slow on the benchmarks. It did well in the code-size tests, producing a remarkably small .EXE file in the minmain( ) test. The library had no fgetc( ) nor fputc( ). These should have at least been defined as macros in stdio.h. We appreciated the fact that EcoSoft included the source for CC, its driver program. Given that they had written a fairly good book on C, we expected good documentation from EcoSoft. We were disappointed to find a very short table of contents and no index. We found the documentation difficult to use. The getting-started and compiler-operation sections were too brief. The library functions did not include examples and were not terribly well organized for easy reference. The compiler was hard to use relative to others in our environment without recompilation of the CC driver.

Compilers

Wizard C V. 2.1
Wizard Systems Software
11 Willows Ct.
Arlington, MA 02174
(617) 641-2379

DeSmet C V. 2.4
C Ware Corp.
P.O. Box C
Sunnyvale, CA 94087
(408)720-9696

Lattice C V. 2.14
Lattice
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950

Datalight C V. 1.0
Datalight
11557 8th Ave NE
Seattle, WA 98125
(206) 367-1803

8088 Software Dev Pkg V. 2.08a
c-systems
P.O. Box 3253
Fullerton, CA 92634
(714) 637-5362

Optimizing CI-86 V. 2.20k
Computer Innovations
980 Shrewsbury Ave. #310
Tinton Falls, NJ 07724
(201) 542-5920

C Programming System V. 2.0
Mark Williams Co.
1430 W. Wrightwood
Chicago, IL 60614
(312) 472-6659

Microsoft C V. 3.0
Microsoft
10700 Northup Way
Bellevue, WA 98004
(206) 828-8088

Aztec C86 V. 3.20c
Manx Software Systems
Box 55
Shrewsbury, NJ 07701
(201) 780-4004

DRI C V. 1.1
Digital Research
P.O. Box 579
Pacific Grove, CA 93950
(408) 649-3896

CC-86 V. 2.0
Control C Software
6441 SW Canyon Ct.
Portland, OR 97221
(503) 297-7153

Ecosoft-C V. 2.0
6413 N. College Ave.
Indianapolis, IN 46220
(317) 255-6476

Software Toolworks C V. 3.2
Software Toolworks
15233 Ventura Blvd., #18
Sherman Oaks, CA 91403
(818) 986-4885

Interpreters

Instant C V. 1.00
Rational Systems
P.O. Box

Run/C V. 1.11
Age of Reason
318 E. 6th #123
New York, NY 10003

C-terp V. 1.04
Gimpel Software
3207 Hogarth Lane
Collegeville, PA 19426
(215) 584-4261

Table 6
Addresses of Compiler and Interpreter Vendors

Lattice C

The Lattice compiler had some strengths and weaknesses (e.g., screen I/O) in execution speed. Although one benchmark is inadequate to assess a compiler's optimization, it should be noted that Lattice won the optimization test hands down. Lattice didn't do particularly well on the code-size tests, but it is one of the few vendors providing the flexibility of an integer-only version of printf( ). Lattice also allows small programs to be changed to .COM files and provides several large-memory models and an option that allows the programmer to trade some pointer-arithmetic facility for better generated code.

The LC program could not handle directory path names in its file list, even though LC1 and LC2 allowed this. The math library was required to do any floating-point stuff, not just transcendentals, etc. Also, the math library had to precede the std library in the search order. The manual documented neither of these facts. Having long supported C for MSDOS, Lattice has produced above average documentation for its compiler and included a good index into the functions. While the installation and getting-started section was brief, it was adequate. The compiler-operation section was good, but finding the utilities operations was a little difficult; these should have had their own sections. The function descriptions were fairly well done but lacked examples. The function descriptions did include a caution paragraph, which is good. This was the only vendor to supply a packing list with the package.

Mark Williams C Programming System

The Mark Williams compiler was the fastest compiler in the screen I/O tests. MWC provides an integer-only version of printf( ). We couldn't determine from the manual what the memory model options were when using MWC object format (−vsmall and −vlarge also force Microsoft object format). There appeared to be no way to specify the desired memory model with MWC object format, implying that the small model was the only one available. An attempt to cast a long to a char led to the following compiler message:

```
113: At 591: Fatal error: no match, op = 62
```

This was not too helpful, and the C syntax *was* legal. And there was no list of errors in the manual, with or without probable cause.

This compiler had a fairly well done manual, with a sample program to compile and run during installation.

C Programmers:

Consider 104 Ways To Be More Productive

If you find and choose the right development software, you can: cut development effort, make impractical projects feasible, and eliminate unproductive, frustrating aspects of programming.

Confused? We'll help you sort thru the huge number of alternatives. Call for comparisons or information.

We carry 27 C Compilers, 4 C Intepreters, 49 Support Libraries, 5 C source debuggers, and 19 other C Add-ons for programming with MSDOS, Macintosh, or CP/M — more than 104 products, really. Here are some of the best products available:

Learn C Programming Only $95

"Introducing C" Interpreter

Computer Innovations has done it again! This interactive implementation is combined with a full screen editor and a thorough, self-paced manual.

You can develop programs faster by getting immediate feedback. Programs will start instantly upon your command. There is no need to wait for "compile and link."

Introducing C includes demo programs, powerful C language interpreter, complete C function library, full screen editor, color graphics, and C language compatibility. PCDOS $95

Inventive Programming Becomes Possible with 300 + ESSENTIAL, tested, fast, rountines to Rely On.

C Utilities Library by Essential Software

Recent Enhancements to Graphics, Windows, AT Support

Every application you write is likely to require functions were you feel like you are "reinventing". Don't. Even if you use only 5% of this library, you will come out ahead on schedule and cost.

Full business Graphics, Window support, polled Communications, and Data Entry support have recently been added/upgraded along with more functions for DOS Interface and AT support. String handling, screen control, "word processor" functions, memory management, directory and path access, date handling, program chaining, keyboard and printer control are traditional strengths.

Full source code is included. No royalties are charged to include functions in your programs. 95% are C for portability and to make it practical for you to understand or modify them.

Lattice, Microsoft, C86, Mark Williams, Aztec, Desmet and Wizard C are supported. Specify which you need.

Substantial time, effort, testing and attention has been invested by Essential Software developing, documenting and supporting this comprehensive library. Make new projects practical and interesting. Use this tested and reliable library.

Some functions are PC-specific. Most support any MSDOS. $159.

SORT/MERGE Files for Clean, Fast Maintenance

with OPT-TECH SORT

Performance should not suffer with DOS or other "free" sorts. ISAMs alone are slow when 10% or even less is changed/added.

OPT-TECH includes:

- CALLable and Standalone use
- C, ASM, BAS, PAS, FTN, COBOL
- Variable and fixed length
- 1 to 9 fields to sort/merge
- Autoselect of RAM or disk
- Options: dBASE, BTrieve files
- 1 to 10 files Input
- No software max for # Records
- All common field types
- By pass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds. MSDOS. $90.

Which Compiler Features Do You Need?

Optimizing C86 Compiler

Over the years the Optimizing C86 has evolved to be the most complete set of C compiler tools. It includes utilities, a rich library, and thorough tech support. In line 8087/287 routines run up to 100 times faster than the 8086 math package. The source code to all routines is included, so you have complete control over how they work. Thorough ROM support, Intel UDI & VMS cross versions are available.

More of the features you want include:

- special IBM-PC library • 2 math and 2 I/O libraries
- full memory utilization of the 8086/88/186/286
- compatibility with most commercial libraries
- Version 2.3 has support for source level debuggers

MSDOS $339

Fast File Access with Source

C-Index +

C-Index + contains a high performance ISAM, balanced B + Tree indexing system with *source* and *variable length* fields. The result is a complete data storage system to eliminate tedious programming and add efficient performance to your programs.

Features include random and sequential data access, virtual memory buffering, and multiple key indexes.

With *no royalties* for programs you distribute, full source code, and variable length fields C-Index + fits what you are likely to need.

Save time and enhance your programs with C-Index +. MSDOS $375

File Management: MultiUser/MultiLanguage

BTRIEVE

Billions and billions of bytes! That's what you can control with Btrieve's file manager. Btrieve gives you the ISAM capability you need without the maintenance headaches.

Using b-trees for optimum performance, Btrieve automatically maintains your files in sorted order on up to 24 different fields. And Btrieve offers you the fastest search algorithm available, to give you instantaneous access to any individual record. You can locate any record in 4 disk reads or less (thanks to Btrieve's RAM cache, usually less). With Btrieve you can stop wasting your time being a file clerk and concentrate on more productive tasks.

Btrieve's other features include:

- 4 gigabyte file size
- 4090 byte record length
- 255 byte key length
- duplicate, modifiable, and null keys
- up to 24 key indexes per file
- automatic file recovery after power failure

Btrieve's Local Area Network version lets you migrate your software to multiuser environments without changing your code. And offers you multiuser update capability beyond simple file locking schemes. Available *for all programming languages* as well as C. MSDOS.Single user $245. Multiuser $595.

Btrieve. Don't settle for less.

Call for details, comparisons, or for our "C Extras Packet" with over 50 pages of information about C support products.

THE PROGRAMMER'S SHOP

The programmer's complete source for software, services and answers

128-LC Rockland Street, Hanover, MA 02339 (617) 826-7531 (800) 421-8006

Ask about COD and PO's. All formats available. Prices subject to change. Names of products and companies are generally their trademarks.

Circle **no. 72** on reader service card.

The function descriptions were reasonably well done and readable. There could have been more examples, and many functions were packed into one description, making it hard to extract pertinent information.

Microsoft C

The Microsoft compiler was fast in nearly all categories, although it didn't do well on the random-access diskio( ) test. We found it to be a carefully designed compiler, very well documented, and with some well-thought-out features. It provides an integer-only version of printf( ) and a variety of large-memory models, plus mixed-model programming. This last feature and 8087 emulation are two features only Microsoft offers.

Microsoft's compiler set a very high standard for software documentation. Two full binders were included: a complete language reference manual plus the compiler and library reference manual. The accompanying slip case folded down to form a book stand. We found it difficult to fault any aspect of Microsoft's documentation package, except for the slight jitter in their laser printer. One experienced C programmer in the group noted that he had never spent more than a couple of minutes with the documentation, because he was able to find what he needed very quickly.

The Microsoft compiler includes the Bessel functions; has excellent Unix/Xenix portability; and is very easy to use.

C Interpreters

In addition to the compilers, we evaluated three C interpreters: C-Terp, Instant C, and Run/C. We were fairly demanding of the interpreters, and our comments should probably be taken as an experienced programmer's view. A beginner might be happier than we were with these products. They were generally faster than the compilers when it came to making a change and seeing what the results were, and that's important when you're learning a language. We did, however, have some complaints.

All three forced some rewriting of our benchmark programs. All three lacked some run-time support modules, in particular fread and fwrite. (C-Terp supplied them, but not in the standard release versions.) All three were memory-hungry, requiring some 300K to execute. We found at least one bug in each of the interpreters in the process of running our benchmarks, once we got the benchmark code in a form each compiler would accept. Of the three, Run/C was the least expensive, but also the least flexible. Block comments had to start in column one with only the "/*" on the line; the terminating "*/" had to meet similar constraints. Run/C was also slow in loading and executing. When we specified a program from the command line, Run/C loaded it, then checked for errors; when one was found, Run/C aborted interpretation, and we lost the (significant) load time. Run/C could not handle pointers to functions.

Instant C was fastest and had most of the features we felt we couldn't get along without,but we found that its editor got in the way. We were frustrated by its limitation to two array indices and by its small work area. It had nice, APL-like immediate-mode variable output that was very handy during the debug phase.

C-Terp was the most flexible and came closest to implementing the complete language. It had the ability to interface to functions written in assembly language, although requiring a compatible compiler to implement the interface. Assembly-language interface is promised for the near future for the other two interpreters. Unfortunately, C-Terp's documentation was sparse, a problem for beginners.

These tools are useful for checking out a short algorithm quickly, but not as the sole language implementation for developing complex pieces of software. You can learn C from them, although a purist might argue that you don't know C until you have worked with the full language. Of these interpreters only C-terp offered one of C's greatest strengths, the flexibility of using pointers as data.

There are three questions to ask yourself in considering buying one of these interpreters or recommending it to others: Can your programming needs be met by an interpreter alone? If you're planning to develop significant software in C, they probably can't. Is it worthwhile to you to purchase both an interpreter and a compiler? It may well be; they address different needs. And if you intend to buy an interpreter, which one? We've provided some material here for the beginning of your evaluation process, but note that we only looked at three C interpreters. And this is an experienced programmer's view of these products. For the benchmark results see Table 8 (below).

Interpreter:	tint	tlong	array	pointer	looptst	opt	fibtest	sieve
C-terp	.2	.2	22.1	28.0	41.7	126.4	54.5	329.2
Instant C	<0.1	<0.1	<0.1	<0.1	.8	1.6	19.2	4.4
Run/C	2.4	2.5	115.2	113.1	205.3	815.5	252.5	407.3

Table 8
Addresses of Compiler and Interpreter Vendors

THE PROGRAMMER'S SHOP™

helps save time, money and cut frustrations. Compare, evaluate, and find products.

SERVICES

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 700 products

Free Literature - Compare Products

Evaluate products Compare competitors Learn about new alternatives One *free call* brings information on just about any programming need Ask for any ***"Packet"*** or "Addon Packet" ☐ ADA, Modula ☐ "AI" ☐ BASIC ☐ "C" ☐ COBOL ☐ Editors ☐ FORTH ☐ FORTRAN ☐ PASCAL ☐ UNIX/PC or ☐ Debuggers, Linkers, etc

RECENT DISCOVERIES

PC LINT - full C program checking and big, small model. C86, Lattice. MSDOS $ 95

ARTIFICIAL INTELLIGENCE

ARITY/PROLOG-full, debug, to ASM&C, 16 Meg use, windows, strings. With compiler $1950. MSDOS $495

ExpertEASE - Expert system tool. Develop by describing examples of how you decide. PCDOS $625

Expert LISP - Interpreter: Common LISP syntax, lexical scoping, toolbox, graphics. Compiler. 512K MAC $465

EXSYS - Expert System building tool. Full RAM, Probability. Why, serious, files PCDOS $275

GC LISP - "COMMON LISP", Help. tutorial, co-routines, compiled functions, thorough. PCDOS Call

M Prolog - full, rich, separate work spaces. MSDOS $725

PROLOG-86 - Learn fast. Standard, tutorials, samples of Natural Language. Exp. Sys. MSDOS $125

TLC LISP - "LISP-machine"-like. all RAM, classes, turtle graphics 8087 CPM-86. MSDOS $235

WALTZ LISP - "FRANZ LISP" - like, 611 digits, debugger, large programs. CPM80 MSDOS $159

MicroProlog - improved MSDOS $235

BASIC

ACTIVE TRACE, DEBUGGER - BASICA, MBASIC, interactive, well liked MSDOS $ 79

CADSAM FILE SYSTEM - full ISAM in MBASIC source. MSDOS $150

BASCOM-86 - Microsoft	8086	279
CB-86 - DRI	CPM86, MSDOS	419
Data Manager - full source	MSDOS	325
InfoREPORTER - multiple	PCDOS	115
Prof. Basic - Interactive, debug	PCDOS	89
TRUE BASIC - ANSI	PCDOS	125

Ask about ISAM, other addons for BASIC

EDITORS FOR PROGRAMMING

BRIEF Programmer's Editor - undo, windows, reconfig. PCDOS $195

VEDIT - well liked, macros, buffers, CPM-80-86. MSDOS. PCDOS $119

C Screen with source	86/80	75
Epsilon - like EMACS	PCDOS	195
PMATE - powerful	8086	185
XTC - multitasking	PCDOS	95

COBOL

Microsoft Version II - upgraded. Full Lev. II, native, screens. MSDOS $500

Dig Res-decent	MSDOS 525
Macintosh COBOL - Full.	MAC 1850
MBP - Lev II, native, screen	MSDOS 885
MicroFocus Prof.-full	PCDOS call
Ryan McFarland-portable	MSDOS 695

C LANGUAGE

C-terp Interpreter by Gimpel, full K&R, .OBJ and ASM interface. 8087 MSDOS $275

INSTANT C - Interactive development - Edit. Source Debug, run. Edit to Run - 3 Secs. MSDOS $445

"INTRODUCING C" - Interactive C to learn fast. 500 page tutorial. examples, graphics PCDOS $ 95

Wizard C - Lattice C compatible, full sys. III syntax, lint included, fast, lib. source. MSDOS $450

MSDOS C86-8087, reliable	call
Lattice C - the standard	call
Microsoft C 3.0 - new	279
Williams - debugger, fast	call
CPM80 - EcoPlus C-faster, SLR	275
BDS C - solid value	125

MEGAMAX C - native Macintosh has fast compile, tight code, K&R. toolkit, .OBJ, DisASM MAC $275

MACINTOSH Hippo II	375
Consulair's MAC C, toolkit	395

Compare. evaluate. consider other Cs

C ADDONS

APPLICATION TOOLKIT by Shaw - Complete: ISAM, Screen, Overlay mgnt, report gen. Strings, String math. Source. CPM, MSDOS $395

COMMUNICATIONS by Greenleaf ($159) or Software horizons ($139) includes Modem7, interrupts, etc. Source. Ask for Greenleaf demo.

C SHARP Realtime Toolkit-well supported, thorough, portable, objects, state sys. Source MANY $600

CIndex + -full B+Tree, vari. length field. Source, no royal. MSDOS $369

dbVista FILE SYSTEM - full indexing, plus optional record types, pointers. Source, no royal. MSDOS $450

CHelper: DIFF, xref, more	86/80	135
CTree - source, no royal	ALL	369
dBC ISAM by Lattice	8086	229
Greenleaf-200 +	MSDOS	159
OTHER: C Utilities by Essential	MSDOS	129
PHACT-up under UNIX, addons	MSDOS	225
ProScreen - windows	PCDOS	275
SCREEN: CURSES by Lattice	PCDOS	125
Software Horizons - Blocks I	PCDOS	139
Turbo V - Greenleaf C, fast	PCDOS	159
Windows for C	MSDOS	175

FORTRAN LANGUAGE

MacFORTRAN - full '77, '66 option. toolbox, debugger, 128K or 512K. ASM-out option MAC $375

RM/Fortran - Full '77. BIG ARRAYS. 8087, optimize, back trace, debug. MSDOS $525

Ask about Microsoft, Supersoft, others.

MS FORTRAN-86 - Improved.	MSDOS	239
DR Fortran-86 - full '77	8086	249
PolyFORTRAN-XREF, Xtract	PCDOS	165

LANGUAGE LIBRARIES

MultiHALO Graphics-Multiple video boards, printers, rich. Animation, engineering business. Any MS language, Lattice C86 $195, for Turbo $95.

Screen Sculptor - slick, thorough, fast, BASIC, PASCAL. PCDOS $115

GRAPHMATIC - 3D, FTN, PAS	PCDOS	125
File MGNT: BTrieve - all lang.	MSDOS	215
Micro: SubMATH - FORTRAN full	86/80	250
MetaWINDOW - icons, cup	PCDOS	139
PANEL - many lang. term	MSDOS	249

OTHER LANGUAGES

ASSEMBLER-ask about Turbo ASM ($95), ED/ASM ($95) - both are fast, compatible. or MASM ($125), improvements.

BetterBASIC all RAM, modules. structure. BASICA - like PCDOS $185

SNOBOL4 + -great for strings, patterns. CPM86, MSDOS $ 85

MacASM - full, fast, tools	MAC	115
Assembler & Tools - DRI	8086	159
PL1-86	8086	495
PCFORTH - well liked	MSDOS	95

SUPPORT PRODUCTS

PLINK - a program-independent overlay linker to 32 levels for all MS languages. C86 and Lattice. $315

Multilink - Multitasking	PCDOS	265
Pfinish - Profile by routine	MSDOS	345
Polylibrarian - thorough	MSDOS	95
PolyMAKE	PCDOS	95
ZAP Communications-VT100. TEK 4010, full xfer	PCDOS	65

DEBUGGERS

Periscope Debugger - load after "bombs", symbolic, "Reset Box", 2 Screen, own 16K. PCDOS $279

Advanced Trace 86 Symbolic	PCDOS	149
Atron Debugger for Lattice	PCDOS	395
CODESMITH-86 - debug	PCDOS	129
PFIX-86 Debugger	MSDOS	169
TRACE86 debugger ASM	MSDOS	115

Call for a catalog, literature, and *solid* value

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339

Mass: 800-442-8070 or 617-826-7531 8517

Note: All prices subject to change without notice
Mention this ad. Some prices are specials.
Ask about COD and POs. All format s available
UNIX is a trademark of Bell Labs.

Circle **no. 97** on reader service card.

54 Circle no. 15 on reader service card.

Software Toolworks C

The Software Toolworks compiler for MSDOS is a port of their C80 compiler for CP/M. The original compiler was a descendant of Small C, adding such features as multi-dimensional arrays and structures. The MSDOS version does not currently support the long, float, or double data types. Longs and floats will be available in an optional "mathpack," currently under development. The lack of these data types meant that we could not run some benchmarks, and the prtf( ) benchmark had to be modified so that the same number of lines were produced, but all longs, floats, and doubles were changed to ints.

Although the compiler does not support longs, it does support fseek for files in excess of 64K. The file position has to be passed as an array of two ints. As a result, a special version of the diskio benchmark was written; the principal changes were to make all longs arrays of int and to use a file created with the Lattice compiler to obtain the random position to seek to. The seek addresses were identical to those used in other benchmarks. The compiler also did not support initialization of data of class auto. Several of the benchmarks had to be changed to account for this feature.

Other unimplemented features included: typedef, bit fields, and line control (#line). Major restrictions included: function calls had to have the same number of arguments as the called function definition, which resulted in a kludge in printf, fprintf, sprintf, scanf, fscanf, and scanf; declarations were allowed only at the start of a function; #define could not have arguments. The printf kludge required that stdio.h be included for minprtf. The compiler supported the small model only (64K of code and 64K of data). The compiler included a function profiler that recorded the total number of calls to a function and the time spent in each function. In contrast to the C80 compiler, this compiler did not support in-line assembly code, although details of the assembly language interface were provided in the manual. This was clearly a subset compiler, with poor documentation. There was an index, though

Aztec	$199 – Pro Package $500
Microsoft	$279
Computer Innovations	$395
DRI	$350
Mark Williams	$495
Control-C	$500
c-Systems	$199
Wizard	$450
DeSmet	$109 + $50 (debugger) + $35 (DOS LINK)
Datalight	$60
Ecosoft	$49.95
Lattice	$425
Software Toolworks	$49.95 + ($29.95 float)
C-terp	$300
Instant C	$495
Run/C	$149.95

Table 7
Compiler and Interpreter Prices

there were many functions to a page, making it difficult to find and read the descriptions. The function descriptions were terse and devoid of examples, again making it difficult for a new user.

Wizard C

Wizard's was generally a fast compiler. It supports a variety of large-memory models, lets the programmer trade some pointer-arithmetic capabilities for better generated code, and allows very small programs to be changed to .COM files. Wizard's got the highest marks for support.

Field width specification did not work for character conversions in printf( ); e.g., printf(%17c,c) produced one character, not 17. Careful reading of K&R leaves us with small doubt that Wizard is in the wrong. It is certainly safe to say that if Wizard maintains that field width is not supposed to work with %c, then they stand alone.

The Wizard compiler was an obvious port down from Unix, as the documentation reflected Unix throughout. While the getting-started section was small (one page) and no installation section was included, we appreciated the 22 pages of error messages. The function descriptions were readable, but like functions were grouped together and there were no examples.

Excellent diagnostics were provided for portable code; to increase portability from Unix, the library included Unix functions that have little meaning but are a pain to remove (e.g., getpid). The compiler supports anachronisms (=+ . . .); supports the use of libraries for other compilers by changing call and return specifics; and includes lint. It has PL/M compatibility and is very easy to use.

Miscellaneous Comments

Microsoft, Wizard, Mark Williams, Control C, Ecosoft

These all had drivers that allowed the compilation of several files (assembly of assembly language files) and optional linkage of the files. This proved to be a great boon.

Mark Williams, Control C

Excellent source level debuggers and very nice examples of usage in the manual and on disk; does not interfere with screen I/O.

DeSmet, Aztec, c-systems

Included source or symbolic level debuggers, but we did not test them.

Microsoft, Lattice, Wizard

Support use of third party or optional source level debuggers, such as SYMDEB.

Aztec, Wizard, Lattice, DeSmet

Provide for generation of ROMable code.

Microsoft, Aztec, Mark Williams, Control C, c-systems, DeSmet, Wizard

The first six used environment to pass "usual" options to compiler; Wizard used a file and this approach was slightly preferred. DeSmet and c-systems had bugs in usage.

Microsoft, Aztec, Wizard, Contol C, Mark Williams

Support new structures (passing, assigning, and returning) and enum and void.

Aztec, DRI

Could not locate math functions in table of contents; includes overlay support.

Mark Williams, Control C

Unable to determine from the documentation what the specifics of the large model were.

Aztec, CI-86, Wizard, Software Toolworks

Library source included or available.

Summary

Were we objective? We all had our preferences and unequal experience with the compilers going into the review; we can't expect that we entirely overcame them. But large amounts of data can swamp out fairly extreme priors, and it was hard to deny the numbers we were getting.

So, who won? If you pressed us to declare a winner, we'd probably say the Microsoft or Aztec compiler. But the Wizard compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested. And the Mark Williams compiler had the best debugger. Our data would support different results, depending on how you chose to weight the features we assessed. No two programmers will weight them the same way. And we took almost no account of price. For addresses of compiler vendors see Table 6 (page 54); for compiler prices see Table 7 (page 50).

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 191.**

Circle **no. 134** on reader service card.

A Peephole Optimizer for Assembly Language Source Code

by David L. Fox

The need for small fast programs is a constant of computer programming. Unfortunately, small fast programs tend to be difficult to write and maintain. As a result a programmer often has to make a choice between assembly language and a high-level language.

D. E. Cortesi, who has discussed this problem in several "Dr. Dobb's Clinic" columns,[1] has pointed out that an optimizing compiler is a powerful incentive to choose a high-level language. The compilers currently available for microcomputers, however, fall far short of the state of the art in terms of the size and speed of the code they produce. In fact, many of these compilers produce code that could be improved by the simple technique of peephole optimization.

In this article, I describe a standalone peephole optimizer that can post-process the output of any compiler producing assembly language source code. The program is written in C and has been compiled with Software Toolworks' C/80 compiler. According to a published benchmark comparison,[2] in terms of both code size and execution speed this is one of the better C compilers available for CP/M. Nevertheless, peephole optimization can improve its output, and I will use it for the examples in this article.

David L. Fox, 2140 Braun Dr., Golden, CO 80401

Program Optimization

The term *optimization,* as it is usually applied to programs, is actually something of a misnomer: it derives from the word *optimum,* meaning the best possible, and no program optimizer or optimizing compiler can seriously claim to produce the best possible code. A more realistic goal is merely to improve the code.

The greatest improvement in a program usually comes from using a better algorithm. If you are not convinced of the importance of choosing an efficient algorithm, I suggest that you write two routines to sort an array of numbers, one using a bubble sort and another using a Shell sort. Use them to sort an array of 1000 numbers. The bubble sort, which requires a time proportional to the square of the number of items sorted, will run tens or even hundreds of times longer than the Shell sort, which requires time proportional to $n^{1.5}$ to sort n items. Unfortunately, the choice of algorithm is not something that we can easily mechanize. If you want a substantial improvement in the performance of a program, rewriting it using a better algorithm may be the only choice available.

Algorithms for finding and replacing regular expressions are well known. Common inefficiencies in compiler output can be specified as regular expressions. So . . .

The more modest improvements obtained by the means discussed below should not be dismissed, however, because they frequently lend themselves to implementation by a program (an optimizing compiler or a separate optimizer) and require little

effort on the part of the programmer. Many of these improvements involve recognizing specific constructions in the input and replacing them with more efficient code in the output. This process is easily mechanized.

One rich source of potential improvement in compiler output is removing the recalculation of values that are already available. For example, a straightforward compilation of a statement such as

```
a[i] = b[i] + c[i]
```

will calculate the byte offset associated with the subscript *i* three times. (If a, b, and c are arrays of 16-bit integers, this offset is *2*i.*) Substantial savings would be possible if this type of situation were recognized and the offset calculated only once.

An even simpler example of this is illustrated in Listing One (page 68) where the compiler produces the pair of instructions:

```
SHLD i
LHLD i
```

After execution of SHLD i, the contents of i are still in the HL register pair and the LHLD i may be omitted. Note, however, if the LHLD instruction is labeled, it may not be omitted.

Another example in Listing One is the following:

```
MOV A,H
ORA L
JNZ L1
LXI H,0
```

The zero flag is set if HL contains zero, in which case the jump to L1 is not taken and the LXI H,0 may be removed.

Simple compilation of conditionals and loops often results in multiple jump instructions that can be easily improved. A jump to a jump can be replaced by a single jump instruction, as shown in Listing One. Another commonly found construction is the jump over a jump, such as

```
    JZ  L1
    JMP L2
L1: . . .
```


How to go from UNIX to DOS without compromising your standards.

It's easy. Just get an industry standard file access method that works on both.

C-ISAM™ from RDS.

It's been the UNIX™ standard for years (used in more UNIX languages and programs than any other access method), and it's fast becoming the standard for DOS. Why?

Because of the way it works. Its B+ Tree indexing structure offers unlimited indexes. There's also automatic or manual record locking and optional transaction audit trails. Plus index compression to save disk space and cut access times.

How can we be so sure C-ISAM works so well?

We use it ourselves. It's a part of INFORMIX,® INFORMIX-SQL and File-it!,™ our best selling database management programs.

For an information packet, call (415) 424-1300. Or write RDS, 2471 East Bayshore Road, Palo Alto, CA 94303.

You'll see why anything less than C-ISAM is just a compromise.

RDS™

RELATIONAL DATABASE SYSTEMS, INC.

© 1985, Relational Database Systems, Inc. UNIX is a trademark of AT&T Bell Laboratories. INFORMIX is a registered trademark and RDS, C-ISAM and File-It! are trademarks of Relational Database Systems, Inc.

Circle no. 49 on reader service card.

Add EDITING to your Software with CSE Run-Time™

Your program can include all or a portion of the *C Screen Editor* (CSE).

CSE includes all of the basics of full screen editing plus source in C for only $75. For only $100 more get CSE Run-Time to cover the first 50 copies that you distribute.

Use capabilities like Full cursor control, block move, insert, search/replace or others. Portability is high for OSes, terminals, and source code.

Call for the "CSE Technical Description" and for licensing terms and restrictions.

Full Refund if not satisfied in first 30 days.

Call 800-821-2492

Solution Systems

335-D Washington Street
Norwell, MA 02061
617-659-1571

Circle **no. 75** on reader service card.

Scrap your LINKER with FASTER C

Reliably:

CUT Compile times (by 15% to 55%)
CUT Testing times (by 12% to 37%)

HOW: FASTER C keeps the Lattice C or C86 library and any other functions you choose in memory. It manages a jump table to replace the LINKER and immediately execute your functions. You can also CALL active functions interactively to speed your program debugging. It includes many options for configuration and control.

"Automatic" support for new libraries by reading the .OBJ files makes support for new libraries quick and simple.

AVAILABLE FOR PC-DOS, IBM-AT, AND ANY 256K MSDOS SYSTEM.

ONLY $95.
Full Refund if not satisfied during first 30 days.
Call **800-821-2492**

Solution Systems™

335-D Washington St., Norwell, Mass. 02061
617-659-1571

Circle **no. 93** on reader service card.

C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing and manipulating C programs. Use C HELPER's UNIX-like utilities which include:

DIFF and **CMP** – for "intelligent" file comparisons.
XREF – cross references variables by function and line.
C Flow Chart – shows what functions call each other.
C Beautifier – make source more regular and readable.
GREP – search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

C Helper is written in portable C and includes both full source code and executable files for $135 for MS-DOS, IBM AT CPM-80 or CPM-86. Use VISA, Master Card or COD.

Call: **800-821-2492**

Solution Systems™

335 Washington Street
Norwell, MA 02061
617-659-1571

Circle **no. 94** on reader service card.

PROLOG-86™

Become Familiar in One Evening

Thorough tutorials are designed to help learn the PROLOG language quickly. The interactive PROLOG-86 Interpreter gives immediate feedback. In a few hours you will begin to feel comfortable with it. In a few days you are likely to know enough to modify some of the more sophisticated sample programs.

Sample Programs are Included like:

- **an EXPERT SYSTEM**
- **a NATURAL LANGUAGE INTERFACE** (it generates a dBASE II "DISPLAY" command)
- **a GAME** (it takes less than 1 page of PROLOG-86)

PROTOTYPE Ideas and Applications QUICKLY

Serious development of experimental systems and prototypes is practical with the full syntax of PROLOG-86. 1 or 2 pages of PROLOG is often like 10 pages in "C".

Programming Experience is not required but a logical mind is. PROLOG-86 supports the de facto STANDARD.

RECENT IMPROVEMENTS: Access to MSDOS, on-line help, load Editor.

AVAILABILITY: All MSDOS, PCDOS systems.

FREE with order: "Best of Prolog-86 Programs" — contest entries include: a primate expert system, an automobile expert system, a blocks world natural language system, etc. Call before November 30.

Only $125
Full Refund if not satisfied during first 30 days.
800-821-2492

Solution Systems

335-D Washington Street
Norwell, MA 02061
617-659-1571

Circle **no. 95** on reader service card.

which may be replaced by

```
    JNZ  L2
L1: ...
```

Any unlabeled instruction following an unconditional jump can never be executed and may be removed. Also candidates for removal are unreachable instructions created by the jump optimization described above. For example:

```
    JMP  L1
    .
    .
    JMP  L2
L1: JMP  L3
```

After we replace JMP L1 with JMP L3 and remove the label L1, JMP L3 becomes unreachable and may also be removed. Thus, one improvement can create an opportunity for further improvement, making it worthwhile to pass the program through the optimizer more than once.

The examples so far have not involved any trade-offs between code size and execution speed. In the following situation, speed or size can be optimized, but not both. Pointer dereferencing (obtaining the value stored at a given address) may involve the following subroutine:

```
deref:  MOV A,M
        INX  H
        MOV H,M
        MOV L,A
        RET
```

Excluding the RET instruction, the above subroutine is only four bytes long. Because any program that uses pointers or arrays does quite a bit of dereferencing, use of the subroutine deref will produce a smaller program. If speed is more important, however, replacing the three-byte instruction CALL deref with four bytes of in-line code will result in faster execution. Some compilers allow the user to choose whether smaller or faster code is more important, but most compilers for microcomputers do not provide this option.

The details of the compiler and the machine instruction set may offer other opportunities for improvement. One common possibility arises from the fact that many compilers for CP/M produce 8080 code. If you have a Z80 processor, you may find ways to replace combinations of 8080 instructions with Z80 instructions. A good example is a 16-bit subtraction, which requires six instructions on an 8080 but only a single Z80 instruction. Identifying the possible improvements for your compiler will require careful study of its output and runtime library.

These examples illustrate the types of improvements that the peephole optimizer can make. Many other improvements, which require some knowledge of the organization and structure of the program such as recognizing constant expressions and removing them from loops, are beyond the scope of the optimizer presented here.

If the optimizer is specific to one compiler, it can be quite simple; see the optimization section of Small-C version 2.[3] On the other hand, if you want to use the program with different compilers or have some flexibility in what changes to make, the optimizer will require most of the capa-

Swap diskettes with popular CP/M* computers!

Just $69.95 turns one of your PC's floppy drives into a CP/M computer "look-alike" with UniForm-PC.

Imagine a software breakthrough that gives your IBM PC, PC-XT, PC-AT or compatible the ability to *directly* read, write and format diskettes from most popular CP/M computers—8 or 16 bit! Remarkable UniForm-PC actually reconfigures your floppy drive to emulate the selected CP/M format, allowing your applications programs and utilities to directly access data files that were previously out of reach.

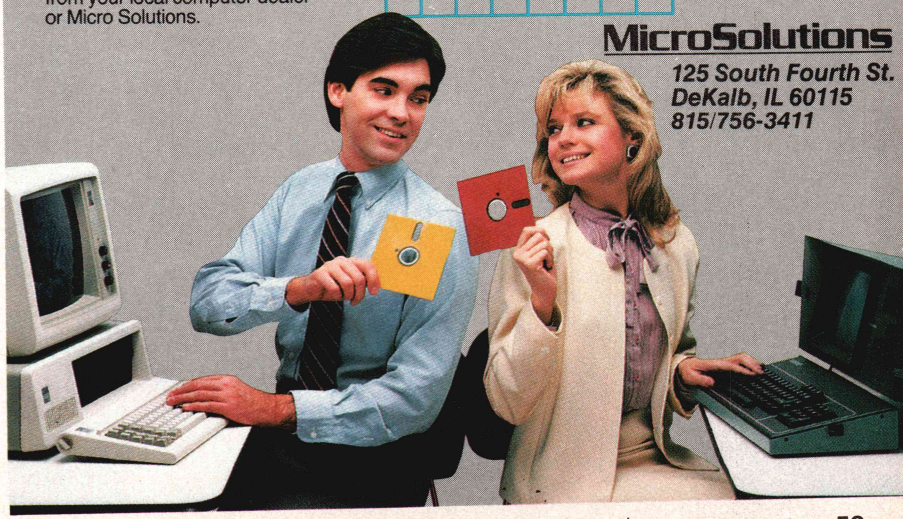
Menu-driven UniForm-PC is easy-to-use and inexpensive. Simply load, select the proper diskette format and go! DOS procedures are unchanged when you use the CP/M diskette. You can even start a project on a PC at work and finish it on a CP/M machine at home without the need for additional hardware or modifications! At just $69.95, CP/M compatibility never cost so little!

UniForm-PC is available now from your local computer dealer or Micro Solutions.

For CP/M computer owners, *UniForm* bridges the gap between non-compatible CP/M formats, as well as providing access to MS-DOS** files. It's also just $69.95.

Trademarks:
*Digital Research
**Microsoft Corporation

MicroSolutions
125 South Fourth St.
DeKalb, IL 60115
815/756-3411



Circle **no. 74** on reader service card.

The C for Microcomputers

AZTEC

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

Manx Aztec C86

"A compiler that has many strengths . . . quite valuable for serious work"

Computer Language review, February 1985

Great Code: Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhrystone benchmark (CACM 10/84 27:10 p1018) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster, Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/ Link Time
Dhrystone Benchmark			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

Great Features: Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler	Symbolic Debugger
AS86 Macro Assembler	LN86 Overlay Linker
80186/80286 Support	Librarian
8087/80287 Sensing Lib	Profiler
Extensive UNIX Library	DOS, Screen, & Graphics Lib
Large Memory Model	Intel Object Option
Z (vi) Source Editor -c	CP/M-86 Library -c
ROM Support Package -c	INTEL HEX Utility -c
Library Source Code -c	Mixed memory models -c
MAKE, DIFF, and GREP -c	Source Debugger -c
One year of updates -c	CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

Aztec C86-c Commercial System	**$499**
Aztec C86-d Developer's System	**$299**
Aztec C86-p Personal System	**$199**
Aztec C86-a Apprentice System	**$49**

All systems are upgradable by paying the difference in price plus $10.

Third Party Software: There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

C-tree $395	**Greenleaf $185**
PHACT $250	**PC-lint $98**
HALO $250	**Amber Windows $59**
PRE-C $395	**Windows for C $195**
WindScreen $149	**FirsTime $295**
SunScreen $99	**C Util Lib $185**
PANEL $295	**Plink-86 $395**

MACINTOSH, AMIGA, XENIX, CP/M-68K, 68k ROM

Manx Aztec C68k

"Library handling is very flexible . . . documentation is excellent . . . the shell a pleasure to work in . . . blows away the competition for pure compile speed . . . an excellent effort."

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C	Creates Clickable Applications
Macro Assembler	Mouse Enhanced SHELL
Overlay Linker	Easy Access to Mac Toolbox
Resource Compiler	UNIX Library Functions
Debuggers	Terminal Emulator (Source)
Librarian	Clear Detailed Documentation
Source Editor	C-Stuff Library
MacRam Disk -c	UniTools (vi,make,diff,grep) -c
Library Source -c	One Year of Updates -c

Items marked -c are available only in the Manx Aztec C86-c system. Other features are in both the Aztec C86-d and Aztec C86-c systems.

Aztec C68k-c Commercial System	**$499**
Aztec C68d-d Developer's System	**$299**
Aztec C68k-p Personal System	**$199**
C-tree database (source)	**$399**
AMIGA, CP/M-68k, 68k UNIX	**call**

Apple II, Commodore, 65xx, 65C02 ROM

Manx Aztec C65

"The AZTEC C system is one of the finest software packages I have seen"

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS, Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

Aztec C65-c ProDOS & DOS 3.3	**$399**
Aztec C65-d Apple DOS 3.3	**$199**
Aztec C65-p Apple Personal system	**$99**
Aztec C65-a for learning C	**$49**
Aztec C65-c/128 C64, C128, CP/M	**$399**

Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

Manx Cross Development Systems

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST. Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

HOSTS: VAX UNIX ($3000), PDP-11 UNIX ($2000), MS-DOS ($750), CP/M ($750), MACINTOSH ($750), CP/M-68k ($750), XENIX ($750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are $300 to $500 (non VAX) or $1000 (VAX).

Call Manx for information on cross development to the 68000, 65816, Amiga, C128, CP/M-68K, VRTX, and others.

CP/M, Radio Shack, 8080/8085/Z80 ROM

Manx Aztec CII

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."

80-Micro, December, 1984, John B. Harrell III

Aztec C II-c (CP/M & ROM)	**$349**
Aztec C II-d (CP/M)	**$199**
C-tree database (source)	**$399**
Aztec C80-c (TRS-80 3 & 4)	**$299**
Aztec C80-d (TRS-80 3 & 4)	**$199**

How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

How To Get More Information

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing systems. Call for information.

MANX

To order or for information call:

800-221-0440

UNIX is a registered TM of Bell Laboratories. Lattice TM Lattice Inc., C-tree TM Faircom, Inc., PHACT TM PHACT ASSOC., CI Optimizing C86 TM Computer Innovations, Inc., MACINTOSH, APPLE TM APPLE, INC., Pre-C, PLINK 86, TM PHOENIX, HALO TM Media Cybernetics, Inc., C-terp, PC-lint TM GIMPLE Software, Windscreen, SunScreen TM SunTec, PANEL TM Roundhill Computer Systems Ltd., WINDOWS FOR C TM Creative Solutions, XENIX, MS TM MICROSOFT INC., CP/M TM DRI, AMIGA, C64, C128 TM COMMODORE Int.

Manx Aztec C86 is the best C for MS-DOS and you can prove it yourself!

"A compiler that has many strengths ... quite valuable for serious work"
Computer Language review, February 1985

Manx Aztec C86 - The C For MS-DOS

Manx Aztec C86 is clearly the best C software development system for MS-DOS. Aztec C86 is the only C compiler for MS-DOS that provides the level of performance, features, documentation, and support required for serious, professional software development. You can prove it yourself. All you have to do is order Aztec C86 from Manx, evaluate it, and, if you like it, keep it. If you don't like it, send it back within 30 days and we'll cancel your order.

If you keep your Manx Aztec C86, as 99% do, you'll be in with the best company.

Manx Aztec C86 Features:

Optimized C compiler: Unsurpassed for code quality and speed. Optionally generates 80186 and 80286 code. Full K & R.

Symbolic Debugger: Execution trace, break points, display data in floating point, integer, character, or hex format. Evaluate expressions. Detect illegal memory stores, modify memory/registers, disassemble code.

Manx AS86 Macro Assembler: Supports macroes, 8086, 80186, and 80286 instructions in Intel format. Fast execution.

LN86 Overlay Linker: Links small, large, and mixed memory model routines, supports overlays, and options for producing ROM based code.

Librarian: Build and modify personal or system run time libraries.

8087/80287 Sensing Library: One library simulates floating point, another assumes the presence of an 8086 or 80287 math chip, the third senses the existence of a math chip, and if it finds one it uses it.

Profiler: Provides a run time analysis of your code to pinpoint code segments to optimize.

UNIX Library: Compatible with UNIX C. Fast I/O. Terminal I/O can be buffered or unbuffered.

DOS Library: Time and date functions, program forking (exec), program chaining, directory commands, I/O port support, sysint support, BIOS functions, and BDOS functions.

Screen & Graphics Library: Screen and cursor functions. Fast routines for drawing lines, circles, elipses, points, and setting colors.

CP/M-86 Library (-c): Produce programs for CP/M-86.

Large Memory Model: Manx Aztec C86 supports programs and data of any size. Global data has a max size of 64k.

Intel Object Option: Interface to software that requires Intel object format, such as PLINK86.

Z (vi) Source Editor (-c): Fast, powerful editor, Macro capabilities, undo, ctags, buffers for commands and data, and all the bells and whistles that make vi fanatics fanatical.

ROM Support Package (-c): Startup routine, linker options for separate placement of code and data, special utilities like the Intel HEX Utility, documentation, and library source.

Library Source Code (-c): UNIX, screen, graphics, and math function libraries.

Mixed Memory Models (-c): Mix large code and small data, small code and large data, or mix within type.

UniTools (-c): The UNIX utilities make, diff, and grep.

One year of updates (-c): As new versions are released, updates are automatically sent.

Technical Support: Manx has a full time staff to provide support via telephone & bulletin board.

Items marked -c are special features of the Aztec C86-c system.

Manx Aztec C86 is available in four configurations: Manx Aztec C86-c, Manx Aztec C86-d, Manx Aztec C86-p, and Manx Aztec C86-a. The -p and -a systems are not intended for commercial work and do not incorporate the same compilers as the -c and -d systems. All systems are upgradable.

Aztec C86-c Commercial System	**$499**
Aztec C86-d Developer's System	**$299**
Aztec C86-p Personal System	**$199**
Aztec C86-a Apprentice System	**$ 49**

Manx Cross Development Systems

Manx Aztec C compilers are available as native or as cross development systems for PC-DOS, MS-DOS, Macintosh, CP/M-86, CP/M-80, TRSDOS, Apple II, and Commodore 64/128.

Cross development involves two computer systems: the development system (HOST) and the execution system (TARGET). This method is useful when the TARGET machine is slower or more limited than the HOST.

HOSTS: VAX UNIX ($3000), PDP-11 UNIX ($2000), MS-DOS ($750), CP/M ($750), Macintosh ($750), CP/M-68k ($750), XENIX ($750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x85 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

Additional TARGETS are $300 to $500 (non VAX) or $1000 (VAX). Call for information, on cross development to the 68000, 65816, Amiga, C128, CP/M-68K, VRTX, and others.

How To Become a Manx.Aztec C User

Call 1-800-221-0440 or 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

For More Information: call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

Manx maintains a large professional staff to service and support Manx users. You will get fast delivery and great service dealing directly with Manx.

Support Software for Manx Aztec C86

C-tree $395: B-tree database system. Easy to use. Available for Aztec C for MS-DOS, Macintosh, CP/M-86, CP/M-80, and others. Includes source.

PHACT $250: Powerful database system. Available for Manx Aztec C compilers for MS-DOS, CP/M-86, CP/M-80, and Macintosh.

PANEL $295: Create screens via simple, powerful editing commands. Select colors, edit fields. Directly input data to a multi-keyed file utility included with the system.

SunScreen $99: Create and modify formatted screens easily. Validate fields, select colors, create screens for both the color and monochrome cards. With library source SunScreen is $199.

WindScreen $149: Combines SunScreen with a powerful window utility.

Windows for C $195: Versatile window utility that supports IBM PC compatible and some non-compatible environments.

AMBER Windows $99: Powerful, low priced window package.

HALO $250: The ultimate C graphics package. It supports viewports, shapes, and multiple graphics cards. A less expensive version is available for just the PC mono and color cards.

FirsTime $295: Syntax checking while you edit greatly shortens compile time.

Pre-C $395: Powerful Lint-like utility locates structural and usage errors. Easily checks multiple files for bad parameter declarations and other interface errors. Lint users will find the user interface a dream come true.

PC-LINT $98: Lint-like utility that supports large memory models, has clear error messages, and executes quickly, has lots of options and features that you wouldn't expect at this low price.

Greenleaf Functions $185: Source for over 200 C and assembler functions. They are great, they work, they are used extensively, and are economically priced. Clear documentation and easy to use interface round out an impressive package.

C Utility Library $185: C and assembler source for screens, windows, color graphics, asynch communications, and more. The color graphics and speed of this package are impressive.

Plink-86 $395: MS-DOS linkage editor for producing and maintaining overlayed programs. It works with Aztec C86 in Intel object format mode.

30 Day Guaranty:

Any Manx Aztec development system can be returned within 30 days for a refund if it fails to meet your needs. Restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in new condition. Returned items must be received by Manx within 30 days. A restocking fee may be required.

Discounts:

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing C systems.

Manx Aztec C Distribution:

In the USA, Manx Software Systems is the exclusive distributor of Aztec C. Telephone or mail order sales other than through Manx are unauthorized.

Circle **no. 62** on reader service card.

MANX

To order or for information call:

800-221-0440

UNIX is a registered TM of Bell Laboratories. Lattice TM Lattice Inc., C-tree TM Faircom, Inc., PHACT TM PHACT ASSOC., CI Optimizing C86 TM Computer Innovations, Inc., MACINTOSH, APPLE TM APPLE, INC., Pre-C, PLINK 86, TM PHOENIX, HALO TM Media Cybernetics, Inc., C-terp, PC-lint TM GIMPLE Software, Windscreen, SunScreen TM SunTec, PANEL TM Roundhill Computer Systems Ltd., WINDOWS FOR C TM Creative Solutions, XENIX, MS TM MICROSOFT INC., CP/M TM DRI, AMIGA, C64, C128 TM COMMODORE Int.

WHITESMITHS, LTD. 370 COMPILER

NOW YOU CAN RUN BOTH C AND PASCAL ON IBM MAINFRAMES

FEATURES:

- Full implementation of the C programming language for the IBM 370/43XX/30XX
- Supports full ISO Level 0 Pascal, extended to support separate compilation
- Uses standard IBM tools to facilitate debugging in the standard IBM save area layout
- Compatible with the full range of Whitesmiths' compilers (PDP-11, VAX, 68000, 8080, and 8086)
- Provides optional cross support for MS/PC-DOS, CP/M-86, CP/M68k, and CP/M-80
- Runs under all versions of OS (VM, MVS, SVS, MVT, and MFT)
- Includes unlimited use of libraries in binary form
- Runs interactively under TSO and CMS
- Direct EBCIDIC support

CALL WHITESMITHS NOW AT (800) 225-1030

MASS. RESIDENTS CALL (617) 369-8499

WHITESMITHS, LTD.
97 LOWELL ROAD
CONCORD, MA 01742
TELEX 750246

INTERNATIONAL DISTRIBUTORS: **Australia,** Neology, Ltd., No. 1 Rosebery Ave., Rosebery 2018, N.S.W., (790) AA74948; **France,** COSMIC s.a.r.l., 76 Quai Des Carrieres, 94227 Charenton Le Pont, (842) 232507; **Germany,** GEI, Albert-Einstein-Strasse 61, 5100 Aachen Walheim, (841) 8329745; **Japan,** Advanced Data Controls Corp., Kyoritsu, Kojimachi Bldg. 5, Kojimachi 5-Chome, Chiyoda-Ku, Tokyo 102, (781) 32902; **Sweden,** Unisoft AB, Fiskhamnsgatan 10, S-41455 Goteborg, (854) 20120; **United Kingdom,** Real Time Systems Ltd., P.O. Box 70, Douglas, Isle of Man, (851) 628356

Circle no. 108 on reader service card.

bilities of a general text editor. For that reason, I have borrowed heavily from Chapters Five and Six of Kernighan and Plauger's book *Software Tools in Pascal*,[4] which give Pascal listings for a fairly complete editor. Listing Five (page 76) contains the needed routines (translated into C and slightly extended) from Kernighan and Plauger's editor.

Regular Expressions

Once you have identified output from your compiler that may be improved, you must communicate descriptions of the improvable constructions to the optimizer program. The notation must be precise and unambiguous—capable of specifying constructions such as:

```
first line:     SHLD    (any label)
second line:    LHLD    (same label
                         as above)
```

A notation sufficiently powerful to describe such strings is used widely within the Unix environment. This notation, adopted here, is that of regular expressions.

A regular expression is a description of a set of strings. The simplest case is when the regular expression describes a single string: the regular expression is just the string itself. For example, the regular expression for the 8080 store HL instruction mnemonic is SHLD. Nonprinting characters are represented by the standard C escapes:

\b	backspace
\n	newline
\r	carriage return
\t	tab
\\	backslash
nnn	byte with octal value *nnn*

An alternative way of looking at a regular expression is as a pattern. A string matches the pattern if, starting at the left, each character of the string matches the corresponding character of the pattern. Because the optimizer is concerned with identifying lines of assembly language code, we will say that any line containing a string described by a regular expression matches that regular expression.

To specify more complex patterns, we need an element that can match more than one character in the input string. One such useful element is the wild card character (.), which will match any single character. We can also specify one of a group of characters by using a character class, which consists of a [, followed by a list of characters, and ends with a], to match a single occurrence of any of the characters enclosed in the square brackets. For example, [SL]HLD will match SHLD or LHLD.

We may use a shorthand notation for all consecutive letters or digits: [a-z] matches any lower-case letter, and [A-Z0-9] matches any upper-case letter or any digit. If the first character of the character class is ~, it negates the character class: it will match any character *except* those in the list fol-

C Programmers Quit Working So Hard!!

THE GREENLEAF FUNCTIONS™

The GREENLEAF FUNCTIONS GENERAL LIBRARY has over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and systems interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library.**) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions.** Simply the finest C library (and the most extensive). All ready for you.

THE GREENLEAF FUNCTIONS™

The Library of C Functions that probably has just what you need . . . **TODAY!**

- already has what you're working to re-invent
- already has over 200 functions for the IBM PC, XT, AT, and compatibles
- already complete . . . already tested . . . on the shelf
- already has demo programs and source code
- already compatible with all popular compilers
- already supports all memory models, DOS 1.1, 2.0, 2.1
- already optimized (parts in assembler) for speed and density
- already in use by thousands of customers worldwide
- already available from stock (your dealer probably has it)
- It's called the **GREENLEAF FUNCTIONS.**

The Library of C Functions is Waiting for You

Specify compiler when ordering. Add $7.00 for UPS second-day air (or $5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check or P.O. In stock, shipped same day.

- General Libraries $185
- Comm Library $185
- CI C86 Compiler $349
- Lattice C $395
- Mark Williams $475

For Information: 214-446-8641

Prices are subject to change without notice.

GREENLEAF SOFTWARE©

2101 HICKORY DR.
CARROLLTON, TX 75006

Circle **no. 43** on reader service card.

lowing the ~. For example, [~aA] matches any character except a or A.

Character classes are essential to matching constructions such as "any label." The standard C identifier consists of any string of letters, digits, or underscore characters that does not begin with a digit; thus, the character class that matches the first character of an identifier is [a-zA-Z_], and the remainder of the characters match the class [a-zA-Z0-9_].

To fully describe a label with a regular expression, we need a construction for "the remainder of the characters" that does not limit us to a specific number of characters. We achieve this by following any character (or character class) by an asterisk, *. The result, called a closure, will match zero or more occurrences of the single character preceding the *. For example, aa* matches any string consisting of one or more occurrences of the letter a. The regular expression describing a C identifier can now be given as:

```
[a-zA-Z_][a-zA-Z0-9_]*
```

This actually matches an identifier of unlimited length, but it serves our needs.

We now need some way of specifying "same label as above." For this, we draw on the ability to group portions of a pattern and refer to the group(s) later. A portion of a pattern that appears between the pairs of characters \(and \) constitutes a group. A group of characters matched by a parenthetical pattern may be identified by the notation \1 for the first group in the pattern, \2 for the second, up to \9 for the ninth. For example,

```
\(mat\).*\1
```

matches mathematician while

```
\(. . .\).*\1
```

matches barbarian, mathematician, and any string that repeats a group of three letters. The notation developed so far allows us to describe the example at the beginning of this section with two regular expressions:

```
SHLD\t\([a-zA-Z_]
  [a-zA-Z0-9_]*\)
LHLD\t\1
```

Finally, it is sometimes useful to require that the string matching a regular expression occur at the beginning or end of a line. The characters to do this are ^ for the beginning of a line and $ for the end of a line. Thus, ^A matches any line that begins with A, and ^$ matches only empty lines. Any characters that have special meanings (e.g., [ ]*.\^$~-) may have the special meaning turned off by preceding the special character with \. The pattern \.$ matches any line that ends with a period.

This flexible and powerful notation suffices to describe most constructions appropriate for improvement with a peephole optimizer. There are a few traps for the unwary, especially when using closures. Remember that x* matches zero or more occurrences of x, so that the regular expression x* matches any line, even those with no x's at all. When you want to match one or more x's, use xx*. Another point is that any closure matches the

The BASIC idea.

Now it's even better.

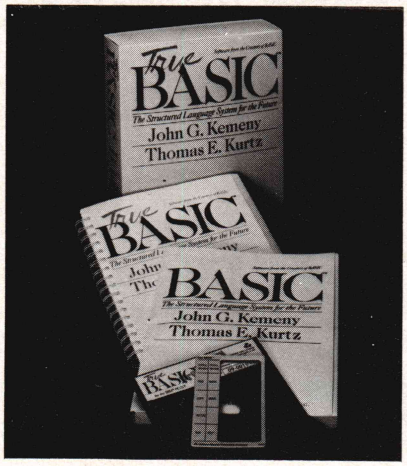
True BASIC™

A new, more powerful version of the world's most widely used programming language–created by the original developers of BASIC, John Kemeny and Thomas Kurtz.

True BASIC is still easy to learn and use yet offers the following advanced features:

- **STRUCTURE**–True BASIC allows you to write modular programs. It supports advanced control structures such as SELECT CASE, IF-THEN-ELSEIF, DO Loops, etc.
- **EXTERNAL PROGRAMS**–Increase your programming efficiency by storing frequently used functions and subroutines in user-defined libraries. True BASIC supports calls to assembly language subroutines.
- **FULL MEMORY**–With True BASIC you can use all the available memory in your computer.
- **SUPERIOR GRAPHICS**–Allow you to draw in your own coordinates, not in terms of pixels. Animation, sound and color are supported.
- **IMPROVED ERROR CHECKING**–Compiler reports syntax errors before running a program. The language incorporates user-defined error messages.
- **WINDOWS**–True BASIC allows you to program multiple screen windows, each with its own parameters.
- **SPEED**–Because True BASIC is compiled, it is faster than interpreted BASICs. Automatic 8087 support is standard.

Now available for the IBM PC. $149.90
Apple Macintosh available Fall, 1985.



True BASIC.™
The BASIC idea made better.

Addison-Wesley
Reading, Massachusetts 01867

True BASIC is a trademark of True BASIC, Inc./IBM is a registered trademark of International Business Machines, Inc./Apple is a trademark of Apple Computer, Inc./Macintosh is a trademark licensed to Apple Computer, Inc.

Circle no. 2 on reader service card.

PROGRAMMER DEVELOPMENT TOOLS

TURBO PASCAL AND UTILITIES

	List	Ours
Turbo PASCAL *ver 3.0* by Borland Int'l ..... *Sale*	70	**49**
Turbo PASCAL *w/8087 or BCD* ..... *Sale*	110	**89**
Turbo PASCAL *w/8087 & BCD* ..... *Sale*	125	**99**
Btrieve by Softcraft	250	**199**
ESP for Pascal by Bellesoft	Call	**Call**
MetaWindows *for Turbo PASCAL* by MetaGraphics .	55	**49**
Multi-Halo Graphics by Media Cybernetics	250	**199**
Screen Sculptor by Software Bottling	125	**109**
Turbo ASYNCH by Blaise Computing	100	**89**
Turbo GRAPHICS TOOLBOX by Borland Int'l.	55	**49**
Turbo POWER TOOLS by Blaise Computing . *New*	100	**89**
Turbo TOOLBOX by Borland Int'l	55	**49**
Turbo TUTOR by Borland Int'l	35	**29**
TurboPower Util w/source by TurboPower Sftwr ..	95	**89**
XTC Text Editor by Wendin	99	**89**

BASIC LANGUAGE

BetterBASIC by Summit Software	200	**169**
8087 Math Support	99	**89**
Run-time Module	250	**239**
Professional BASIC by Morgan Computing	99	**89**
8087 Math Support	50	**47**
True Basic from Addison-Wesley ..... *New*	150	**129**
Run-time Module	500	**459**

OTHER LANGUAGES

8088 Assembler w/Z-80 Translator by 2500 AD ...	100	**89**
APL*PLUS/PC by STSC	595	**449**
Golden Common LISP by Gold Hill	495	**Call**
Janus/ADA by R&R Software	900	**699**
MASM-86 ver 3.0 *w/utilities* by Microsoft	150	**109**
Modula-2/86 by Logitech	495	**439**
MS Fortran 3.3 ..... *Links with MS C 3.0*	350	**239**
MS Pascal 3.3 ..... *Links with MS C 3.0*	300	**219**
Prolog VMS by Automata Design Associates *New*	100	**Call**
Prolog VML by Automata Design Associates *New*	300	**Call**
Prolog VMA by Automata Design Associates *New*	500	**Call**
Prolog-86 by Solution Systems	125	**Call**
RM/Fortran by Ryan-McFarland	595	**439**

OTHER LANGUAGE SUPPORT PRODUCTS

APL2C by Lauer Software ..... *Interfaces APL to C*	150	**139**
Btrieve By SoftCraft	250	**199**
Blaise Tools *for Pascal*	Call	**Call**
FORTRAN Libraries by Alpha Computer Service .	Call	**Call**
Sci Subroutine Lib *for Fortran or Basic*	175	**139**

OTHER PRODUCTS

Advanced Trace-86 by Morgan Computing	175	**149**
Codesmith-86 Debugger by Visual Age	145	**129**
Polytron Products ..... *We Carry a Full Line*	Call	**Call**
Profiler by DWB Associates	125	**89**
Rtrieve by Softcraft	85	**79**
Xtrieve by Softcraft	195	**169**

Periscope Symbolic Debugger
by Data Base Decisions

Write-protect memory board and breakout switch allows instant recovery from runaway code. Provides on-line help, windowing, extensive breakpoints, dual monitor support and more.

List Price $295 Our Price **$269**

PHOENIX PRODUCTS

SUMMERTIME SALE!

In stock and ready for immediate shipping.

Pasm86 *Macro Assembler*	295	**195**
Pfinish *Performance Analyzer*	395	**269**
Pfix-86 Plus *Symbolic Debugger for Plink-86*	395	**269**
Plink-86 *Overlay Linker*	395	**269**
Pmate *Macro Text Editor*	225	**145**
Pmaker *Program Development Manager*	195	**119**
Pre-C *Lint Utility for C*	395	**269**

Dealer Inquiries Invited

C LANGUAGE

C-terp *C Interpreter* by Gimpel Software	300	**269**
C-terp *demo system* ..... *Applied to purchase*	45	**45**
Computer Innovations C-86 Compiler	395	**299**
DeSmet C Compiler with Debugger	159	**145**
Instant C by Rational Systems ..... *Sale*	500	**399**
Lattice C Compiler from Lattice	500	**349**
Lattice C from Lifeboat ..... *Ltd Qty Special*	500	**275**
Mark Williams MWC-86 w/Source Debugger . *Sale*	495	**379**
Safe C Standalone Interpreter by Catalytix	400	**Call**
Wizard C Compiler by Wizard Systems ..... *Sale*	450	**359**
Xenix Development System by SCO	1350	**1099**

MACINTOSH C COMPILERS

DeSmet/Ouye C Compiler ..... *New*	150	**139**
Mac C by Consulair ..... *Call for Details*	**Call**	**Call**
Megamax C *compiler for MacIntosh*	295	**239**

MICROSOFT C 3.0 AND UTILITIES

Microsoft C Compiler version 3.0 ..... *Sale*	395	**259**
Blaise C Tools	125	**109**
Blaise C Tools 2	100	**89**
C-terp by Gimpel Software	300	**269**
C Utility Library by Essential Software	185	**139**
Greenleaf C Functions Library ver 3.0	185	**139**
Greenleaf Comm Library	185	**139**
The HAMMER by OES ..... *New*	195	**179**
Multi-Halo Graphics by Media Cybernetics	250	**199**
PANEL *Screen Designer ver. 6.0* by Roundhill	295	**234**
C Power Paks from Software Horizons	Call	**Call**
Windows for C by Creative Solutions	195	**139**

C UTILITIES

Basic_C Library by C Source	175	**139**
Blaise C Tools	125	**109**
Blaise C Tools 2	100	**89**
Btrieve by SoftCraft	250	**199**
C Power Paks From Software Horizons	Call	**Call**
c-tree by FairCom ..... *New version*	395	**359**
C Utility Library by Essential Software	185	**139**
ESP for C by Bellesoft	Call	**Call**
GraphiC by Scientific Endeavors ..... *New version*	250	**209**
Greenleaf C Functions Library ver 3.0	185	**139**
Greenleaf Comm Library	185	**139**
Multi-Halo Graphics by Media Cybernetics	250	**199**
PANEL *Screen Designer ver. 6.0* by Roundhill	295	**234**
PC Lint by Gimpel Software	100	**89**
Safe C Dynamic Profiler by Catalytix	150	**Call**
Safe C Runtime Analyzer by Catalytix	400	**Call**
Scientific Subroutine Lib *for C* by Peerless	175	**139**
Windows For C by Creative Solutions	195	**139**

The **HAMMER C** Library by OES

This excellent new C library is designed for creating end-user interfaces. There are functions for creating 123-like menus, managing the screen, creating input forms, prompting for inputs and more. No royalties and includes source code.

List Price $195 Our Price **$179**

LATTICE PRODUCTS

All products receive support and updates directly from Lattice. Lattice C is in stock and ready for shipment.

Lattice C Compiler from Lattice	500	**349**
C-Food Smorgasbord	150	**119**
C-Sprite *Program Debugger*	175	**149**
Curses *Screen Manager*	125	**109**
dBC *C Interface for dBase II or III Files*	250	**209**
dBC *with source code*	500	**439**
LMK *Make Facility*	195	**159**
Text Mgmt Utils *(GREP/DIFF/ED/WC/Extract/Build)* ...	120	**105**

TEXT EDITORS

Brief By Solution Systems	195	**Call**
Epsilon *Emacs-like Text Editor* by Lugaru	195	**179**
ESP for C *or* **Pascal** by Bellesoft	Call	**Call**
ESP for C *and* **Pascal** by Bellesoft	Call	**Call**
Vedit by Compuview	Call	**Call**
XTC Text Editor By Wendin	99	**89**

Programmer's Connection

136 Sunnyside Street Hartville, Ohio 44632 (216) 877-3781 (In Ohio)

U.S.; **1-800-336-1166** Canada ; **1-800-225-1166**

Call For Our Catalog

MasterCard VISA

Account is charged when order is shipped
Prices are subject to change without notice

Circle **no. 67** on reader service card.

longest possible string: e.*e matches the entire word *emergence,* not just *eme* or *erge.* Table 1 (below) gives some more examples of regular expressions and the strings they describe. For a more extensive discussion of regular expressions, see Chapter 3 of Aho and Ullman's *Principles of Compiler Design.*[5]

Using the Optimizer

The description of each improvement in the compiler output has two parts: a list of regular expressions describing the code to be replaced and a list of lines of improved assembly language code. A group of lines is replaced only if each line in turn matches the corresponding regular expression. The constructions \1, \2, and so on in a replacement cause the replacement to contain the same string matched by the corresponding "\(\)" group in the list of regular expressions.

The list of potential improvements is placed in a file that is read by the optimizer. Listing Six (page 82) shows such a file for use with C/80 compiler output. Each improvement consists of the following: one or more lines containing regular expressions identifying the construction that may be improved, a separator line that contains a % as its first character, zero or more lines that are to replace the text matched by the regular expressions, and a terminator line starting with %%. The file may contain any number of potential improvements. In Listing Six, comments describe each improvement; they make the output somewhat larger than the input and may be removed if desired.

The program command line syntax is

opt options infile outfile

where infile is the name of the file containing the assembly language source code to be optimized, outfile is the name of the file to contain the optimized output (must be different from infile), and options represents zero or more of the following optional parameters:

- -b*n* Store *n* lines in the internal FIFO buffer. Default is 500.
- -f*file* Read the list of improvements from *file.* Default is opt.dat.
- -j*re* A jump instruction described by the regular expression *re.* Default is \tJ[CEMNOPZ]*\t.
- -l*re* A label described by the regular expression *re.* Default is [a-zA-Z_\.][a-zA-Z_0-9]*.
- -o Omit jump optimization. With this option selected, the optimizer can be used as a general purpose stream editor.
- -u*re* An unconditional jump instruction described by the regular expression *re.* Default is \tJMP\t.
- -v Turn off verbose output. Normally opt displays the last line of every improvable construction on the console; selecting this option omits that output.

^. . .$	Matches any line with exactly three characters
(.*)	Matches any string enclosed in parentheses
/ *[a-zA-Z]**/	Matches any C comment containing only blanks and letters
[~]*	Matches any string of nonblank characters
\([a-z][a-z]*\)\tJMP\t\1	Matches "loop JMP loop" but not "loop1 JMP loop" or "abc JMP xyz"
\(..*\)\1	Matches any string containing two adjacent repeated substrings such as *beriberi, vacuum,* and *tintinnabulation*
\(..*\)\(..*\)\2\1	Matches *affair, accommodate,* and *this is the end* (\1="th", \2="is ")

Table 1
Examples of Regular Expressions and Matching Strings

ROUTINE	CALLS	TICKS	% TIME (Approx.)
amatch	81309	9370	27
omatch	92501	6548	19
fgets	3149	3707	11
outline	3103	3541	10
match	32504	3011	9
locate	36934	2504	7
patsize	42331	2331	7
findsym	1316	1363	4
strcmp	34942	812	2
free	3374	654	2
optimiz	3103	376	1
malloc	3692	287	1
main	1	112	0

Table 2
Profiler Output for Optimizing a Test File

The internal organization of the program is quite simple. Each line is read into a FIFO buffer in memory. The size of this buffer determines the range of jump instructions that may be optimized. The input is compared with the regular expressions of the improvable constructions, which are contained in a linked list. If a match is found, the appropriate replacement is made. A symbol table containing all the labels in the input is maintained for jump optimization.

Because this article is on optimization, it is appropriate to discuss the optimization of the optimizer. As given in Listings Two to Six (pages 68–83), the program processes approximately

200 lines per minute. Using the optimizer on its own assembly language code saved 199 bytes and increased speed by about 2%. Adding to the data file some constructions specific to the optimizer and reoptimizing saved an additional 86 bytes and increased speed by another 1%, for a total improvement of 285 bytes and about 3%. A third run, made with a data file that replaced calls to the pointer dereferencing routine with inline code, increased program size by 473 bytes and speed by 10%.

We might wish to improve two separate aspects of the peephole optimizer's performance. First, consider the amount of improvement the optimizer makes in the programs it processes. We could increase performance in this area by finding more improvable constructions in the compiler output and adding them to the data file opt.dat. Unless we have missed some extremely poor and very common compiler output, however, this approach will likely produce only a small increase in performance. As is usually the case, substantial improvement requires the use of an improved algorithm.

Second, consider the speed of the optimizer in accomplishing its task—presently about 200 lines per minute. Once again, small improvements may be made by tweaking the present code, but major improvements will require a change to better algorithms. By using the profiler included in the Software Toolworks' C/80 package, we can identify those subroutines of the program that are executed most frequently. The profiler output for a run of the optimizer is shown in Table 2 (page 66).

Table 2 shows that the optimizer spends over two-thirds of its time matching regular expressions to strings (match( ) and subsidiary routines). Clearly, any significant increase in speed will require an improvement in the pattern-matching code. I have made a small effort along these lines by replacing the C version of the routine locate( ) with an assembly language version. Because better algorithms for regular expression matching are known, this seems like the most profitable course for increasing the speed of the program. In contrast, replacing the linearly searched linked list used for the symbol table with a more efficient hash table cannot increase the speed of the program by more than 4% because only 4% of the execution time is spent in the routine findsym( ) looking up symbols in the table.

Installation

Although I have endeavored to make the program as flexible as possible so it will be useful with a variety of compilers, I have tested it only with the C/80 compiler. Users of other compilers will have to construct a data file like that in Listing Six for the constructions peculiar to their compiler.

Listings Two to Five are "standard" C and should be portable to other C compilers. My library follows the usage of the Unix (version 7) library and differs only slightly from the library distributed with C/80 and some other C compilers. Listing Seven (page 83) shows the library functions used by the optimizer.

Although I have not tested it, the new Small-C library[6] appears to contain suitable routines, with the exception of the memory management routines malloc( ) and free( ). The optimizer frees blocks of memory without regard to the order in which

Thinking of... 8087, High-Speed Math Support?

THINK C-86 OPTIMIZING C86 FROM COMPUTER INNOVATIONS

The 8087/80287 Math Co-processor chips provide a major leap in the computing capability of the IBM family of personal computers. The fast, accurate, math computational ability provided by these chips not only speeds typical applications but opens a range of applications once reserved exclusively for mini and mainframe systems. The key question is HOW CAN THE C PROGRAMMER TAKE ADVANTAGE OF THE 8087/80287's capabilities? The answer is C86 by Computer Innovations.

EXECUTION SPEED, ACCURACY, IN-LINE CODE

C86 has offered the highest level of 8087/80287 support **since the chips became available**. Specialized libraries make program development a snap. A math function library provides full trig, log and a wide variety of other functions.

But most important, C86 produces **IN-LINE** code. This means that math functions can be performed by the co-processor in immediate succession WITHOUT separate subroutine calls. This eliminates library call "overhead" and results in the highest execution speed attainable.

LIBRARY SOURCE PROVIDES CONFIDENCE - CONTROL

Library Source Code is INCLUDED. - It always has been with C86. This means you have total control over your development environment, and total control means better and faster code.

TECH SUPPORT - GET INTO 8087/80287 PAINLESSLY

Computer Innovations' experienced tech support team is available to get you through the rough spots and provide the assistance/advice/expertise required to get you into 8087/80287 quickly and efficiently. Plus you can take advantage of Computer Innovations' user newsletter, technical notes, and On-Line bulletin board. **No** other Compiler offers more.

CALL FOR FREE BULLETIN

To receive Computer Innovations' free bulletin entitled "How C86 Takes Advantage of Intel 8087/80827 Math Co-processors," or for more information about C86 Call:

800-922-0160

COMPUTER INNOVATIONS, INC.

980 Shrewsbury Avenue,
Tinton Falls, NJ 07724 (201) 542-5920

C86 for all PC-MSDOS computers. $395.00. Visa, Mastercard, personal check and corporate P.O.'s accepted.

Circle **no. 96** on reader service card.

they were allocated. The C/80 library (version 3) also contains suitable routines, provided that the error return value is changed to 0 (NULL) from -1. Listings for suitable memory management routines are also given in Chapter 8 of Kernighan and Ritchie.[7]

Availability

I will return a copy of the source code and an executable .COM file (for CP/M 2.x) to anyone who sends me a postage-paid disk mailer, a formatted 5¼-inch disk, and $10.00. I can write a large number of single-sided, soft-sectored formats but do not have the hardware for hard-sectored or 8-inch disks.

References

[1]D. E. Cortesi. *Dr. Dobb's Journal.* No. 83 (November 1983) p. 10, No. 87 (January 1984) p. 18, and No. 91 (May 1984) p. 12.

[2]Jim Gilbreath and Gary Gilbreath. *Byte.* (January 1983) p. 283.

[3]J. E. Hendrix. *Dr. Dobb's Journal.* No. 74 (December 1982) p. 16, and No. 75 (January 1983) p. 48.

[4]Brian W. Kernighan and P. J. Plauger. *Software Tools in Pascal.* Addison-Wesley, 1981. See also Brian W. Kernighan and P. J. Plauger's *Software Tools* (Addison-Wesley, 1976), which gives the same tools in Ratfor, a Fortran-based language.

[5]Alfred V. Aho and Jeffrey D. Ullman. *Principles of Compiler Design.* Addison-Wesley, 1979.

[6]James Hendrix and Ernest Payne. *Dr. Dobb's Journal.* No. 91 (May 1984) p. 50, and No. 92 (June 1984) p. 56.

[7]Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language.* Prentice-Hall, 1978.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 192.**

Peephole Optimizer

(Text begins on page 56)

Listing One

```
int i,j,m,n;
while (i == 0)
{       i = j + n;
        m = i + j;
        if (j == 1)
                j = 0;
}
```

(a) Fragment of C code

```
                        ; start of loop
.f:     LHLD    i       ; evaluate i == 0
        MOV     A,H
        ORA     L
        JNZ     .g      ; jump out of loop if not true
        LHLD    j       ; i = j + n
        XCHG
        LHLD    n
        DAD     D
        SHLD    i
        LHLD    i       ; m = i + j
        XCHG
        LHLD    j
        DAD     D
        SHLD    m
        LHLD    j       ; evaluate j == 1
        DCX     H
        MOV     A,H
        ORA     L
        JNZ     .h      ; jump if not true
        LXI     H,0     ; j = 0
        SHLD    j
.h:     JMP     .f      ; loop
.g:     DS      0       ; continue with rest of program
```

(b) Assembly language compiler output
(Comments added)

```
                        ; start of loop
.f:     LHLD    i       ; evaluate i == 0
        MOV     A,H
        ORA     L
        JNZ     .g      ; jump out of loop if not true
        LHLD    j       ; i = j + n
        XCHG
        LHLD    n
        DAD     D
        SHLD    i       ; redundant LHLD deleted
        XCHG
        LHLD    j
        DAD     D
        SHLD    m
        LHLD    j       ; evaluate j == 1
        DCX     H
        MOV     A,H
        ORA     L
        JNZ     .f      ; JNZ to JMP improved
                        ; LXI H,0 removed following in line test
        SHLD    j
        JMP     .f
.g:     DS      0       ; continue with rest of program
```

(c) Optimized assembly language code

End Listing One

Listing Two

```
/* optdefs.h -- Definitions and declarations for peephole optimizer. */

/****************************************
*    Copyright 1984 by David L. Fox
*    All rights reserved.
*    Permission is granted for unlimited personal, non-commercial use.
****************************************/

/* Standard definitions (stdio.h) */

#define EOF     -1
#define NULL    0
#define TRUE    1
#define FALSE   0
#define MAXSTR  256     /* Maximum string length */
#define FILE    int     /* File descriptor type */

/* Definitions for optimizer */

#define MAXLINES        500     /* Lines of source in memory  */
#define JTSIZE          10      /* Entries in jump list table */
#define LBLLEN          15      /* Maximum label length */
#define OPTIONS         "B:F:J:L:UU:V"  /* Legal options */
#define LBLTERM         ':'     /* Character which terminates a label */
#define SYMLEN          8       /* Symbol (label) length */
```

```
#define UNDEF           0       /* Undefined symbol */
#define INBUF           1       /* Symbol defined and in buffer */
#define OUTBUF          2       /* Symbol written to output, removed from
                                   buffer */
#define UNJMPD          3       /* Symbol is label of JMP, replaced with
                                   destination */

/* Structure declarations */

struct lkline {         /* Linked list of lines */
        struct lkline *nextln;  /* Link to next line */
        char *ln;               /* This line */
};

struct trfmn {          /* Transformation from old to new code */
        int nold;       /* Number of lines in old construction */
        int nnew;       /* Number of lines in new construction */
        struct lkline *old;     /* Linked list of old reg. exprs. */
        struct lkline *new;     /* Linked list of new substitutions */
        struct trfmn *nexttr;   /* Link to next transformation */
};

struct jmp_lst {        /* List of lines with jumps to a label */
        int njmp;
        int jlines[JTSIZE];     /* Line nos. containing jump to label */
        struct jmp_lst *jlnext;
};

struct symt {           /* Symbol table entry */
        char sname[SYMLEN+1];
        int lineno;             /* Line number in buffer */
        char scode;             /* Status: undefined, in buffer,
                                   written out, replaced */
        struct jmp_lst *jmplst;
        struct symt *snext;
};

/* Definitions for regular expression code */

#define MAXPAT  (2*MAXSTR)
#define CLOSIZE 1       /* Size of a closure entry */
#define CLOSURE '*'
#define BOL     '^'
#define BOLSTR  "^"     /* String consisting of BOL */
#define EOL     '$'
#define ANY     '.'
#define CCL     '['
#define CCLEND  ']'
#define NEGATE  '~'
#define NCCL    '!'     /* Cannot be the same as NEGATE. */
#define LITCHAR 'c'

#define ENDSTR  '\0'
#define NEWLINE '\n'
#define ESCAPE  '\\'
#define DASH    '-'

#define GRPST   '('     /* Codes for grouping and subpatterns */
#define GRPEND  ')'
#define GRPPAT  '#'
#define DITTO   -1

/* #define      Z80             /* Insert this define if you have
                                   a Z80 processor. */
```

End Listing Two

Listing Three

```
/* optglob.c -- Global variable declarations for optimizer. */

/*****************************************
*    Copyright 1984 by David L. Fox
*    All rights reserved.
*    Permission is granted for unlimited personal, non-commercial use.
*****************************************/

char **linep;           /* Array of pointers to lines */
int nlines = 0;         /* Number of lines */
int frstln;             /* Index of oldest line in buffer */
int curln;              /* Current line in buffer */
FILE infd;              /* Input file descriptor */
FILE outfd;             /* Output file descriptor */
unsigned maxlines =     /* Lines in buffer */
        MAXLINES;
int verbose = TRUE;     /* Verbose output flag */
int jmpopt = TRUE;      /* Flag for jump optimization */
int optind = 0;         /* Index of next argument */
char *optarg;           /* Pointer to option argument */
struct trfmn *trans;

char *patfile = "opt.dat\0\0\0\0\0\0\0";        /* File for optimization
                                                   data */
char *lbl = "[a-zA-Z_\.][a-zA-Z0-9_]*";         /* Label template */
char *jmpins = "\tJ[CEMNOPZ]*\t";               /* Jump instruction
                                                   template */
char *ucjmp = "\tJMP\t";                /* Unconditional jump */

char *lblpat, *jmppat, *ucjpat;         /* Pointers to patterns made
                                           from reg. expr. */

struct symt frstsym = {
        ('\0'),         /* sname */
        -1,             /* lineno */
```

(Continued on next page)

JDR DELIVERS QUALITY FOR LESS!

MODEM FOR APPLE OR IBM
$69.95 SPECIFY APPLE OR IBM
FCC APPROVED
* BELL SYSTEMS 103 COMPATIBLE
* 300 BAUD
* AUTO-DIAL/AUTO-ANSWER
* DIRECT CONNECT
* INCLUDES ASCII PRO-EZ™ MENU DRIVEN SOFTWARE
* INCLUDES AC ADAPTOR

DISKETTE FILE
$8.95
WITH PURCHASE OF 50 DISKETTES
$9.95 IF PURCHASED ALONE
HOLDS 70 5¼" DISKETTES

NASHUA DISKETTES
5¼ SOFT SECTOR,
DS/DD
$.89ea QTY 250
BULK PACKAGED
$.95ea QTY 100
$.99ea QTY 50

ORDER TOLL FREE
800-538-5000
800-662-6279 (CA)

TAXAN RGB VISION III
SUPER HI-RES RGB MONITOR
ORIGINALLY MADE FOR ACORN COMPUTER
• 12" SCREEN
• 640 x 262 PIXELS
• SAME SPECS AS MODEL 420
$299.95
NO C.O.D. ORDERS PLEASE

4164-200 9/$10.50
8087-3 $129.00

DISK DRIVES
FOR IBM
TEAC FD-55B $89.95
HALF HEIGHT DS/DD
MPI-B52 ..... $89.95
FULL HEIGHT DS/DD
TANDON TM100-2 ..... $99.95
FULL HEIGHT DS/DD

FOR APPLE
BAL-525
$119.95
* ½ HEIGHT ALPS MECHANISM
* 100% APPLE COMPATIBLE
* ONE YEAR WARRANTY

BAL-500
$139.95
* TEAC — ½ HEIGHT DIRECT DRIVE
* 100% APPLE COMPATIBLE
* 40 TRACK WHEN USED WITH OPTIONAL CONTROLLER
* ONE YEAR WARRANTY

CONTROLLER CARD $49.95
IIc ADAPTOR CABLE $19.95

JDR Microdevices
1224 S. Bascom Avenue, San Jose, CA 95128
800-538-5000 • 800-662-6279 (CA) • (408) 995-5430
FAX (408) 275-8415 • Telex 171-110
© Copyright 1985 JDR Microdevices

Peephole Optimizer (Listing continued, text begins on page 56)

Listing Three

```
        0,              /* scode */
        NULL,           /* jmplst */
        NULL};          /* snext */

struct sym *symtab = &frstsym;          /* Start of symbol table */

/* Global variable declarations for regular expression code */

char pat[MAXPAT];       /* Space for pattern */
int lparen, rparen;     /* Parenthesis counts */
char *gstart[10];       /* Pointers to start of grouped text */
char *gend[10];         /* Pointers to end of grouped text */
char *lgp;              /* Pointer to start of last group */
```

End Listing Three

Listing Four

```
/*  opt.c -- Assembly language optimizer */

/****************************************
*   Copyright 1984 by David L. Fox
*   All rights reserved.
*   Permission is granted for unlimited personal, non-commercial use.
****************************************/

/*
Performs peephole and jump optimization of an assembly language
source file.  Transformations for peephole optimizer are specified
in file opt.dat as follows:

        regular expressions
        for matched lines
        %
        replacement lines
        %%

Command line syntax is

opt [ -bnn -ffile -o -lre -jre -ure -v ] infile outfile

where the options are

-bnn    Use nn lines in the FIFO buffer (default=1000).
-ffile  Read peephole optimization information from file instead
        of opt.dat.
-o      Do not do jump optimization.
-lre    Labels defined by regular expression re.
-jre    Jump instructions defined by regular expression re.
-ure    Unconditional jump instruction defined by regular
        expression re.
-v      Verbose output off.
*/


#include        "b:optdefs.h"
#include        "b:optglob.c"
#include        "b:restuf.c"

/* Main program */

main(argc, argv)
int argc;
char **argv;
{
char inline[MAXSTR], *s, *lines;
int c, i;
char *p, *q;
FILE patfd;
char *fgets(), *newline(), *makepat(), *strcpy();
struct sym *symp, *chksym();

/* Initialization */
```


SCIENTIFIC/ENGINEERING PC GRAPHICS TOOLS

GRAFMATIC™ PLOTMATIC™ for FORTRAN/PASCAL PROGRAMMERS

The GRAFMATIC (screen graphics) and companion PLOTMATIC (pen plotter) libraries of modular scientific/engineering graphics routines let you easily create 2D and 3D plots in customized or default formats. Pen plot preview with GRAFMATIC. Plot interactively or in deferred mode. **Primitives** (mode, color, cursor, character, pixel, line, paint. . .) plus auto-scaling, auto-axis generation, auto-tic mark labeling, **function** plots, **tabular** plots, error bars, auto-function plots (complete plot in default format with one easy call), auto-tabular plots, **log/parametric/contour** plots, 3D rotation/scaling/translation, **wire frame** model (for old time's sake), **hidden line removal** for solid models (GRAFMATIC only), cubic and bicubic spline interpolants, least squares fits, bar and pie charts, screen dump. . . . You name it. We have it! Best of all, the clearest and most complete documentation to be found in microcomputerland. User support? Of course, call us!

SCREEN and PEN PLOTTER Software Libraries for the IBM PC

MICROCOMPATIBLES

301 Prelude Dr.
Silver Spring, MD 20901
(301) 593-0683

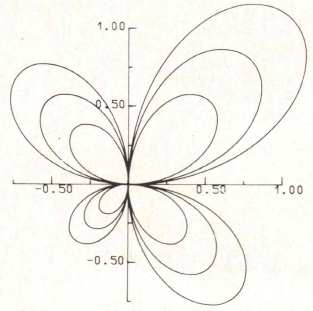
GRAFMATIC*	$135
PLOTMATIC*†	$135
BOTH	$240
OMNIPLOT [S]	$195
OMNIPLOT [P]†	$195
BOTH	$295

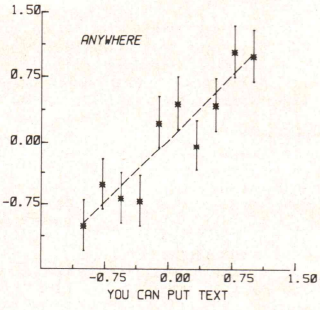
*Specify Compiler (Mfg. and Version): IBM/MS/IBM Prof. FORTRAN
†Specify Plotter: HP/HI/IBM

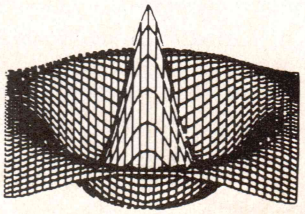
OMNIPLOT [S] OMNIPLOT [P] *NO Programming Required*

Integrated stand-alone graphics libraries to drive your CRT monitor or your pen plotter. Key in data in response to menu prompts or read your data from a disk file. Choose from an assortment of graphics formats: **tabular, line, bar** or **pie charts. Contour plots.** (YES! Just part of our integrated OMNIPLOT library, not an expensive individual item). Create 3-D plots with a choice of **wire frame** or **hidden surface removal** for added realism. Choose **standard, semi-log** or **log-log** scales; gridding; error bars; line colors and types; marker symbol colors and types. Cubic spline interpolations and least squares fitting options. As with our other professional packages we offer clear and careful documentation filled with examples, and user support.

Ask for OMNIPLOT [S] for screen graphics and OMNIPLOT [P] for the pen plotter software library.







Circle no. 61 on reader service card.

```
frstln = 0;
curln = maxlines - 1;

/* Create patterns for jumps and labels from regular expressions */
/* Allocate memory. */
if ((lblpat = sbrk(MAXSTR)) == -1 ||
    (jmppat = sbrk(MAXSTR)) == -1 ||
    (ucjpat = sbrk(MAXSTR)) == -1)
        error("Not enough memory");

/* Create patterns. */
p = lblpat;
makepat(lbl, '\0', lblpat, &p);
p = jmppat;
makepat(jmpins, '\0', jmppat, &p);
p = ucjpat;
makepat(ucjmp, '\0', ucjpat, &p);

/* Allocate memory for groups. */
for (i = 1; i < 10; ++i)
        gstart[i] = sbrk(MAXSTR);
if (gstart[10] == -1)
        error("Not enough memory");

/* Process options. */
while ((c = getopt(argc, argv, OPTIONS)) != EOF)
{       switch(c | 0x20) {
        case 'b':       /* Set FIFO buffer size. */
                maxlines = atoi(optarg);
                break;
        case 'f':       /* Set data file name. */
                strcpy(patfile, optarg);
                break;
        case 'l':       /* R.e. for labels */
                p = lblpat;
                makepat(optarg, '\0', lblpat, &p);
                break;
        case 'j':       /* R.e. for jumps */
                p = jmppat;
                makepat(optarg, '\0', jmppat, &p);
                break;
        case 'o':       /* Omit jump optimization. */
                jmpopt = !jmpopt;
                break;
        case 'u':       /* R.e. for unconditional jumps */
                p = ucjpat;
                makepat(optarg, '\0', ucjpat, &p);
                break;
        case 'v':       /* Switch verbose output. */
                verbose = !verbose;
                break;
        default:
                error("usage: opt [ -bnn -ffile -lre -jre -ure -v ]\
 [ infile [ outfile ] ]");
        }
}

/* Read optimization data file. */
patfd = mustopen(patfile, "r");
gettmpl(patfd);
fclose(patfd);

/* Allocate memory for FIFO buffer. */
if ((linep = sbrk(maxlines*sizeof(char *))) == -1)
        error("Not enough memory");

/* Set up i/o channels. */
outfd = stdout;
infd = stdin;
if (argc > optind)
{       infd = mustopen(argv[optind++], "r");
        if (argc > optind)
                outfd = mustopen(argv[optind++], "w");
}

/* Main loop */
while ((s = fgets(inline, MAXSTR, infd)) != NULL)
{       newline(s);
        /* Update symbol table. */
        if(jmpopt && (symp = chksym(curln)) != NULL)
                unjump(symp);   /* Remove jumps to JMP. */
        /* Optimize current line. */
        optimize(curln);
}
fclose(infd);

/* Output contents of FIFO buffer. */
while (frstln != curln)
        outline();
outline();
if (outfd != 0)
        fclose(outfd);
}

/* newline -- Add a line to buffer, output some old ones to make space
              if needed.  Update frstln, curln, and nlines.
*/

char *
newline(str)
char *str;
{
char *s, *strcpy(), *malloc();

/* Check for buffer wrap around. */
if (++curln >= maxlines) {
        curln = 0;
}
while (nlines >= maxlines || (s = malloc(strlen(str)+1)) == NULL)
{       /* Make some space */
        outline();
}
++nlines;
return(linep[curln] = strcpy(s, str));
}

/* optimize -- Look for improvable code and make improvements */

optimize(ln)
int ln;
{
struct trfmn *p;
int n, m;
char *s;
struct lkline *o;

/* Loop through all possible transformations. */
for (p = trans; p->nexttr != NULL; p = p->nexttr) {
        /* Check lines in buffer against r.e. in transformation. */
        lparen = rparen = 0;
        for (n = p->nold, o = p->old;
           n > 0 && match(linep[(m=ln-n+1) >= 0 ? m : m+maxlines],
           o->ln); --n) {
                o = o->nextln;
```

(Continued on next page)

"C/80 . . . the best software buy in America!"

—MICROSYSTEMS

Now available in MS-DOS

Other technically respected publications like *Byte* and *Dr. Dobb's* have similar praise for **The Software Toolworks' $49.95** full featured **'C' compiler** for CP/M® and HDOS with:

- I/O redirection
- command line expansion
- execution trace and profile
- initializers
- Macro-80 compatability
- ROMable code
- and much more!

"We bought and evaluated over $1500 worth of 'C' compilers . . . C/80 is the one we use."

— Dr. Bruce E. Wampler
Aspen Software
author of *"Grammatik"*

In reviews published worldwide the amazing **$49.95 C/80** from **The Software Toolworks** has consistently scored at or near the top — even when compared with compilers costing ten times as much!

The optional **C/80 MATHPAK** adds 32-bit floats and longs to the C/80 3.0 compiler. Includes I/O and transcendental function library all for only **$29.95!**

C/80 is only one of 41 great programs each **under sixty bucks.** Includes: LISP, Ratfor, assemblers and over 30 other CP/M® and MSDOS programs.

For your **free** catalog contact:

The Software Toolworks
15233 Ventura Blvd., Suite 1118,
Sherman Oaks, CA 91403 or call 818/986-4885 today!

CP/M is a registered trademark of Digital Research.

Circle **no. 99** on reader service card.

Peephole Optimizer (Listing continued, text begins on page 56)

Listing Four

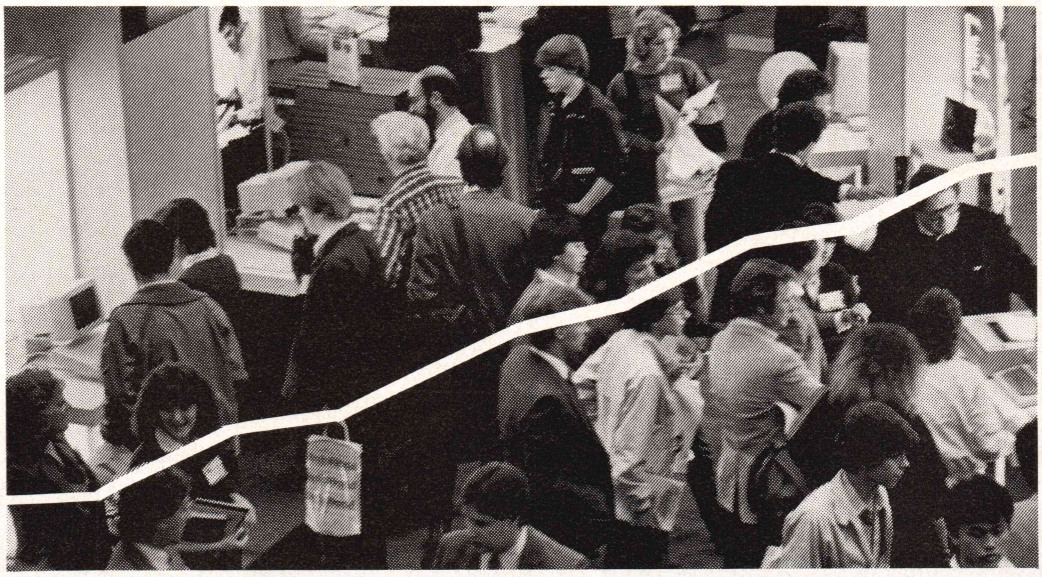
```
        }
        if (n > 0)      /* No match, skip to next transformation. */
                continue;
        if (verbose)    /* Print last line matched. */
                fprintf(stderr, "MATCH---%s", linep[ln]);
        /* Replace matched lines with improved code. */
        if (!subst(ln,p))
                error("substitute failed");
}
}

/* unjump -- Route jumps to jumps directly to target */
/*      syp is pointer to symbol table entry for label on JMP */
unjump(syp)
struct symt *syp;
{
char *p, dpat[MAXSTR];
int i, errflg;
struct symt *dsyp, *isjmp();
struct jmp_lst *jp;

if (syp->scode == OUTBUF) {
        /* Target is not in memory. */

        fprintf(stderr, "%s not in buffer\n", syp->sname);
        for (jp = syp->jmplst; jp != NULL; jp = jp->jlnext)
                free(jp);       /* Release memory used by jump list. */
        errflg = TRUE;
}
else {  /* Target is in memory. */
        errflg = FALSE;
        if ((dsyp = isjmp(linep[syp->lineno])) == NULL)
        {       message(linep[syp->lineno]);
                error("Not a jump");    /* "Can't happen." */
        }

        /* Make literal pattern for substitutions */
        for (p=syp->sname, i=0; *p != '\0'; ++p)
        {       dpat[i++] = LITCHAR;
                dpat[i++] = *p;
        }
        dpat[i++] = CLOSURE;
        dpat[i++] = LITCHAR;
        dpat[i++] = LBLTERM;
        dpat[i] = '\0';
        /* Replace destination label for every entry in jump list. */
        jp = syp->jmplst;
        for (jp = syp->jmplst; jp != NULL && jp->njmp >= 0; jp = jp->jlnext)
        {       for (i=0; i < jp->njmp; ++i)
                {       if (!subline(jp->jlines[i], dpat, dsyp->sname))
                        {       errflg = TRUE;
                                fprintf(stderr,
                                   "Cannot substitute %s in %s\n",
                                   dsyp->sname, linep[jp->jlines[i]]);
                        }
                        /* Add new reference to jump list */
                        addjmp(dsyp->sname, jp->jlines[i]);
                }
                free(jp);
        }
}
syp->jmplst = NULL;
if (!errflg)    /* Delete label from target. */
{       if (!subline(syp->lineno, dpat, ""))
                fprintf(stderr,
                   "Failed to delete %s in %s\n",
                   dpat, linep[syp->lineno]);
        syp->scode = UNJMPD;
}
}

/* I/O routines for optimizer */

/* outline -- Output a line from FIFO buffer. */

outline()
{
struct symt *s, *islab(), *isjmp();
struct jmp_lst *jp;

if (linep[frstln] != NULL) {    /* If line not deleted */
        fputs(linep[frstln], outfd);
        /* Check for labels and jumps. */
        if ((s = islab(linep[frstln])) != NULL)
                s->scode = OUTBUF;
        if ((s = isjmp(linep[frstln])) != NULL)
        {       s->scode = OUTBUF;
                /* Delete jump list. */
                for (jp=s->jmplst; jp != NULL; jp = jp->jlnext)
                        free(jp);
                s->jmplst = NULL;
        }
        --nlines;
        free(linep[frstln]);
}
/* Check for buffer wrap around. */
if (++frstln >= maxlines) {
        frstln = 0;
}
}

/* gettmpl -- Get templates for changes. */

gettmpl(fd)
FILE fd;
{
int n;
struct trfmn *t;
struct lkline *l, **ll, *mm, *newlk();
char line[MAXSTR], *s, *p;
char *fgets(), *makepat(), *makesub(), *strcpy();

/* Allocate memory for first transformation. */
t = trans = (struct trfmn *) sbrk(sizeof(struct trfmn));
t->nexttr = NULL;

while ((s = fgets(line, MAXSTR, fd)) != NULL)
{       lparen = rparen = 0;    /* Initialize parenthesis counters. */
        ll = &t->old;
        t->nold = t->nnew = 0;
        while (*s != '%' && s != NULL)
        {       p = pat;
                if (makepat(s, '\n', pat, &p) == NULL)
                {       message(line);
                        error ("bad r.e.");
                }
                l = newlk(ll, pat);
                ll = &l->nextln;
                ++t->nold;      /* Increment line count. */
                s = fgets(line, MAXSTR, fd);
        }
        if (s != NULL && strncmp(s, "%%", 2) != 0)
        {       mm = &t->new;
                while ((s = fgets(line, MAXSTR, fd)) != NULL &&

                   strncmp(s, "%%", 2) != 0) {
                        /* Get replacement strings. */
                        if (makesub(line, '\n', pat) == NULL)
                        {       message(line);
                                error("bad sub string");
                        }
                        l = newlk(mm, pat);
                        mm = &l->nextln;
                        ++t->nnew;
                }
        }
        /* Allocate memory for next transformation. */
        t->nexttr = (struct trfmn *) sbrk(sizeof(struct trfmn));
        t = t->nexttr;
        t->nexttr = NULL;
}
}

/* newlk -- Create a new lkline structure. */

struct lkline *
newlk(lkp, str)
struct lkline **lkp;    /* Address of pointer to next member of
   linked list. */
char *str;
{
char *strcpy();
struct lkline *l;

if ((l = (struct lkline *)sbrk(sizeof(struct lkline))) == NULL)
        error("memory full");
*lkp = l;
if ((l->ln = sbrk(strlen(str)+1)) == NULL)
        error("memory full");
strcpy(l->ln, str);
l->nextln = NULL;
return(l);
}

/* sttecpy -- Copy string between start and end pointers. */
char *
sttecpy(dest, start, end)
char *dest, *start, *end;
{
char *p;

for (p = dest; start < end; )
        *p++ = *start++;
*p = '\0';
return(dest);
}

/*****************************************/
/* Symbol table routines for optimizer. */
/* addsym -- Add a label to the symbol table. */

struct symt *
addsym(name)
char *name;
{
char *p, *q, *strcpy();
static struct symt *s = &frstsym, *sret;

strcpy(s->sname, name);
```

(Continued on page 74)

A no risk investment guaranteeing high returns is not easy to find,
but that is exactly what you'll get from the

3rd Personal Computer Faire

September 5-7, 1985
Civic Auditorium and Brooks Hall
San Francisco



AN INVESTMENT WITH IMMEDIATE RETURNS

Invest only $10 for one day
$15 for three days

Better yet preregister by August 16, and invest only $12 for three days.

Computer Faire Inc.
181 Wells Avenue, Newton, MA 02159
(617) 965-8350

Name _____
Title _____
Company _____
Address _____
City _____
State _____ *Zip* _____
Phone _____

3DD

Guaranteed returns include:

- **You can compare state of the art microcomputer hardware, software and peripherals from 250 leading companies;**
- **A free comprehensive conference program providing microcomputer solutions for your business, profession and home;**
- **Scores of free hands-on-demonstrations allowing you to try before you buy;**
- **Free one-on-one sessions with professional computer consultants will provide the answers to your specific questions.**

Fill out the coupon for a free prospectus detailing preregistration information, the conference program and exhibiting companies.

An exclusive production of **Computer Faire, Inc.**/A Prentice-Hall Company

Circle **no. 21** on reader service card.

Peephole Optimizer (Listing continued, text begins on page 56)

Listing Four

```
s->lineno = -1;
s->scode = INBUF;
if ((s->snext = malloc(sizeof(struct symt))) == NULL)
        error("symbol table overflow");
sret = s;
s = s->snext;
s->snext = NULL;
s->jmplst = NULL;
return(sret);
}

/* addjmp -- Add line number to list of jumps to label. */

addjmp(name, lineno)
char *name;
int lineno;
{
char *p, *q;
struct symt *symp, *findsym(), *addsym();
int n;
struct jmp_lst **jp;

if ((symp = findsym(name)) == NULL)
{       symp = addsym(name);    /* Symbol not found, add it to table. */
        symp->scode = UNDEF;
}
jp = &symp->jmplst;
/* skip to end of jump list */
while (*jp != NULL && (*jp)->njmp >= JTSIZE)
        jp = &(*jp)->jlnext;    /* Skip to last block of jump list. */
if (*jp == NULL)        /* Allocate space for new block. */
{       if ((*jp = malloc(sizeof(struct jmp_lst))) == NULL)
                error("jump list overflow");
        else    /* Initialize new block. */
        {       (*jp)->njmp = 0;
                (*jp)->jlnext = NULL;
        }
}
(*jp)->jlines[(*jp)->njmp++] = lineno;
}

/* findsym -- Linear search of symbol table. */

struct symt *
findsym(name)

char *name;
{
struct symt *s;

for (s = &frstsym; s->snext != NULL; s = s->snext)
        if (strcmp(s->sname, name) == 0)
                return(s);
return(NULL);
}

/* chksym -- Check for a labeled line or line with a jump. */

struct symt *
chksym(curln)
int curln;
{
char *sttecpy();
struct symt *syp, *s,  *addsym(), *islab(), *isjmp();

        if (syp = islab(linep[curln]))
                syp->lineno = curln;
        if (syp->lineno = -1)   /* First time for this line. */
                syp->lineno = curln;
        if ((s = isjmp(linep[curln])) != NULL)
                addjmp(s->sname, curln);
        if (syp != NULL)
        {       if (match(linep[curln], ucjpat) != NULL)
                        return(syp);    /* Labeled unconditional jump. */
        }
        return(NULL);
}

/* islab -- If str points to a valid label return pointer to its
            entry in symbol table, else return NULL. */

struct symt *
islab(str)
char *str;
{
```


EXTRA!... VALUE and PERFORMANCE with Relocatable Z-80 Macro Assembler FROM MITEK

It's a real bargain! Here's why:

- Only $49.95 plus shipping
- 8080 to Z-80 Source Code Converter
- Generates Microsoft compatible REL files or INTEL compatible hex files
- Compatible with Digital Research macro assemblers MAC & RMAC
- Generates Ditigal Research compatible SYM files
- Full Zilog mnemonics
- INCLUDE and MACLIB files
- Conditional assembly
- Phase/dephase
- Separate data, program, common and absolute program spaces
- Customize the Macro Assembler to your requirements with installation program
- Cross-reference Generation
- Z-80 Linker and Library Manager for Microsoft compatible REL files available as a total package with Macro Assembler for only $95.00 plus shipping
- Manual only is $15

NEW for Turbo Pascal Users

Mitek's Relocatable Z-80 Macro Assembler will also:

- Generate Turbo Pascal in-line machine code include files
- Include files provide Turbo Pascal compatible machine code with assembly language mnemonics as comments

TO ORDER, CALL TOLL FREE: 1-800-367-5134, ext. 804
For information or technical assistance: 1-808-623-6361

Specify desired 5¼" or 8" format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed. Include $5 for postage and handling.

Z-80 is a trademark of Zilog, Inc. MAC, RMAC, and ZSID are trademarks of Digital Research, Inc.

MITEK
P.O. Box 2151,
Honolulu, HI 96805

Circle no. 66 on reader service card.

```
char *p, *q, name[LBLLEN], *amatch(), *sttecpy();
struct symt *syp, *addsym();

p = lblpat;
if ((q = amatch(str, str, &p)) != NULL) {
        if((syp = findsym(sttecpy(name, str, q))) == NULL) {
                        syp = addsym(name);
                }
        }
return(syp);
}

/* isjmp -- If line contains a jump instruction
        return pointer to its destination label symbol table entry
        otherwise return NULL. */
struct symt *
isjmp(line)
char *line;
{
char *p, *q, *r, name[LBLLEN];
struct symt *syp, *findsym();
char *amatch(), *sttecpy();

syp = NULL;
if ((p = match(line, jmppat)) != NULL) {
        q = lblpat;
        if ((r = amatch(p,p,&q)) != NULL) {
                sttecpy(name, p, r);
                if ((syp = findsym(name)) == NULL)
                        syp = addsym(name);
        }
}
return(syp);
}

/**********************************/
/* Utility routines */

/* getopt -- Unpack options from arg list. */
/*       Argc and argv are argument count, and values from main.
        Optstr is a string of legal option letters.
        Getopt returns the next option letter from argv.
        If the letter in optstr is followed by ':' the external
                pointer optarg is set to point to the argument
                following the option letter.
        The external variable optind is set to the index of the
                next argument in the argv array to be processed.
        If an illegal option is found, getopt returns '?'
        If the next argument in argv is not an option
                getopt returns EOF
*/

getopt(argc, argv, optstr)
int argc;
char *argv[], *optstr;
{
static char *cp;
char *p;
char *strchr();

if(optind == 0)
        cp = argv[++optind] + 1;        /* First call */
if(*argv[optind] != '-' || optind >= argc)
        return(EOF);            /* No more options */
if(*cp == '-')
{       ++optind;
        return(EOF);            /* End of options */
}
if((p = strchr(optstr, *cp++)) == NULL)
{       fputs("unknown option: ", stderr);

        putc(*(cp-1), stderr);
        putc('\n', stderr);
        if(*cp == '\0')
                cp = argv[++optind] + 1;
        return('?');
}
if(*(p+1) == ':')
{       if(*cp == '\0')         /* Get argument */
                optarg = argv[++optind];
        else
                optarg = cp;
        cp = argv[++optind] + 1;
}
else
        if(*cp == '\0')
                cp = argv[++optind] + 1;        /* Set up for
                                                   next argv. */
return(*p);
}

/* message -- Send a message to the console (stderr) and return. */
message(s)
char *s();
{
fputs(s, stderr);
putc('\n', stderr);
return(0);
}
```

(Continued on next page)

Release 2.0

C-INDEX™

Variable Length Record Management For C

C-INDEX is a state-of-the-art data management function library for C programmers. Ideal for all data and text based applications. No other package can give you the performance, capability, and portability of C-INDEX. To make sure you are a satisfied customer, we offer a 30 day money-back guarantee. Ask us about it.

- variable length data storage with B+Tree indexing
- high performance, easy to use
- large and small models, fully transportable source
- IBM PC format: Lattice, Microsoft, C86, others
- Macintosh format: Consulair, Manx, others

C-INDEX/FILE ($99) Object code package.

C-INDEX/PRO ($195) Partial source code, no royalties.

C-INDEX/PLUS ($395) Complete transportable source code, no royalties.

Trio Systems

2210 Wilshire Blvd., Suite 289
Santa Monica, CA 90403
213/394-0796

Circle **no. 20** on reader service card.

You read Dr. Dobb's Journal And You Don't Subscribe?!

Save over $23.00 off newstand prices for 2 yrs. Save over $10.00 for 1 yr.

Can you afford to miss an issue with information vital to your interests? As a subscriber you can look forward to articles on Small-C, FORTH, CP/M, S-100, Compiler optimization, Concurrent Programming and more, delivered right to your door. And you'll never miss the issue that covers your project.

YES! Sign me up for ___ 2 yrs. $47 ___ 1 yr. $25

___ I enclose a check/money order
___ Charge my Visa, MasterCard
American Express
___ Please bill me later

Name _____

Address _____

_____

Credit Card _____ Exp. date _____

Account No. _____

Signature _____

This offer good in U.S. only **3043**

Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto CA 94303

Peephole Optimizer (Listing continued, text begins on page 56)

Listing Four

```
/* mustopen -- Attempt to open a file, exit on failure. */

FILE
mustopen(name, mode)
char *name, *mode;
{
FILE fd;
if((fd = fopen(name, mode)) == NULL)
{       fputs(name, stderr);
        error(": cannot open file");
}
return(fd);
}

/* message -- Send a message to the console (stderr) and exit. */
error(s)
char *s;
{
fputs(s, stderr);
putc('\n', stderr);
fclose(stderr);
exit();
}
```

End Listing Four

Listing Five

```
/* restuf.c -- Regular expression routines. */

/* These routines are translations into C of the regular expression
*       matching and pattern generation routines in
*       B.W. Kernighan and P.J. Plauger, "Software Tools in Pascal"
*       chapters 5 and 6.
*   The major extension is the addition of code to allow matching and
*       substitution of parts of expressions using \(...\) for
*       grouping and \n for replacement.
*/

/* match -- Find match anywhere on line. */

match(lin, pat)
char *lin, *pat;
char *amatch();
{
char *pos, *lst, *p;

lst = lin;
for (pos = NULL; *lin != ENDSTR && pos == NULL; ++lin)
{       p = pat;
        pos = amatch(lst, lin, &p);
}
return(pos);
}

/* amatch -- Anchored match */
/* Look for match between lin and pat. */
/* Return pointer to next unmatched character in line,
   or NULL if no match. */
/* lst is pointer to start of line. */
/* lin is pointer to current position in line. */
/* *ppat is pointer to current position in pattern,
   updated if match found. */

char *
amatch(lst, lin, ppat)
char *lst, *lin, **ppat;
{
auto char *lp, *mp, *tp;
auto int level, skip;
char tpat[MAXPAT];
char *strcpy();

while (**ppat != ENDSTR)
{       switch(**ppat) {
        case CLOSURE: {
                *ppat += patsize(*ppat);        /* Step over closure. */
                for (lp = lin; *lp != ENDSTR; )
                        if (! omatch(lst, &lp, *ppat))
                                break;
                if (*(tp = *ppat + patsize(*ppat)) == GRPEND)


        {       skip = patsize(tp);     /* At end of group, skip
                                           over group closure. */
        }
        else
                skip = 0;
        /* lp points to input character that made us fail. */
        /* Match rest of pattern against rest of input. */
        /* Shrink closure by 1 after each failure. */
        while (lp >= lin)
        {       tp = *ppat + patsize(*ppat) + skip;
                if (skip)
                        gend[lparen] = gstart[lparen] +
                           (int)(lp - lgp);
                if ((mp = amatch(lst, lp, &tp)) != NULL)
                        if (skip == 0)
                        {       *ppat = tp;
                                return(mp);
                        }
                        else
                        {       *ppat += patsize(*ppat) + skip;
                                return(lp);
                        }
                else
                        lp--;
        }
        return(NULL);
        /* If mp == NULL failure, else success. */
}
case GRPST: {   /* Start of group */
        mp = *ppat;
        *ppat += patsize(*ppat);
        level = ++lparen;
        strcpy(qstart[level], lin);
        lgp = lin;
        if ((lin = amatch(lst, lin, ppat)) == NULL)
        {       *ppat = mp;
                --lparen;
                return(NULL);
        }
        qend[level] = qstart[level] + (int)(lin - lgp);
        break;
}
case GRPEND: {  /* End of group */
        *ppat += patsize(*ppat);
        ++rparen;
        return(lin);
}
case GRPPAT: {  /* Match previously found group. */
        if ((level = (*ppat)[1] - '0') < 1 || level > 9)
        {       message(*ppat);
                error(" amatch: can't happen");
        }
        *ppat += patsize(*ppat);
        if (gend[level] > gstart[level] + MAXSTR/2)
                return(NULL);   /* Group is too long for tpat. */

                /* Make temporary pattern from matched group. */
                for (tp = tpat, lp = gstart[level]; lp < gend[level]; )
                {       *tp++ = LITCHAR;
                        *tp++ = *lp++;
                }
                *tp = ENDSTR;
                tp = tpat;
                if((lin = amatch(lst, lin, &tp)) == NULL)
                        return(NULL);
                break;
        }
        default:
        if (omatch(lst, &lin, *ppat))
        {       *ppat += patsize(*ppat);
        }
        else {
                return(NULL);
        }
}
}
return(lin);
}

/* omatch -- Match one pattern element at pat. */

omatch(lin, linp, pat)
char *lin, **linp, *pat;
{
char *lpos;
int advance;

advance = -1;
lpos = *linp;

if (*lpos == ENDSTR)
        return(FALSE);
switch(*pat)
{
case LITCHAR:
        if (*lpos == pat[1])
                advance = 1;
        break;
case BOL:
        if (lpos == lin)
                advance = 0;
        break;
case ANY:
        if (*lpos != NEWLINE)
                advance = 1;
        break;
case EOL:
        if (*lpos == NEWLINE)
                advance = 0;
        break;
case CCL:
```

(Continued on page 78)

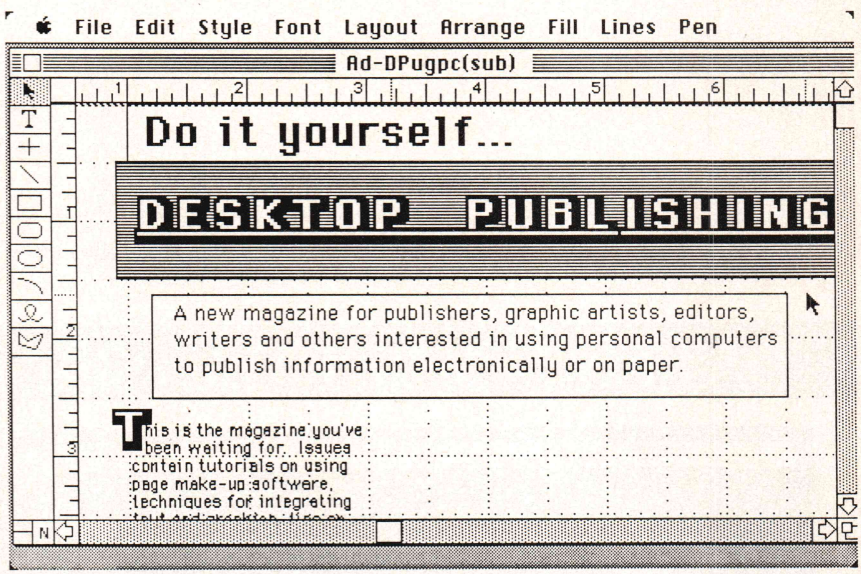
Do it yourself...

DESKTOP PUBLISHING™

A new magazine for publishers, graphic artists, editors, writers and others interested in using personal computers to publish information electronically or on paper.

This is the magazine you've been waiting for. Issues contain tutorials on using page make-up software, techniques for integrating text and graphics, tips on producing more appealing graphics and typography, and helpful information about data communications and electronic publishing.

The bi-monthly magazine covers the latest products for the newly-emerging desktop publishing market, including laser printers, scanning equipment, page make-up software, archival storage devices, and topics of interest to people who do their own electronic or paper publishing.



The cover price is $5, but you can **save $11** by ordering a subscription for $24 and paying with your order. You'll get **seven issues** for the price of six! And there's no risk -- you can cancel after the first issue for a full refund!

We also publish ***User's Guide to CP/M,*** devoted entirely to CP/M computer users, and ***PC USER,*** a new magazine for users of PC-compatible computers and MS-DOS software. Each magazine is $22 for a six-issue subscription. Send payment now and get **seven issues** for the price of six!

☐	DESKTOP PUBLISHING	$24
☐	USER'S GUIDE TO CP/M	$22
☐	PC USER	$22

Name _____

Address _____

City _____

State, Zip _____

MC/Visa _____

Exp. Date _____

2055 Woodside Rd. #180,
Redwood City, CA 94061
415-364-0108 BBS: 415-367-1029

MCI Mail: Tony Bove
CompuServe: 70105,722
The Well: t-c

DESKTOP PUBLISHING: **for anyone who publishes information using computers.**

Circle **no. 106** on reader service card.

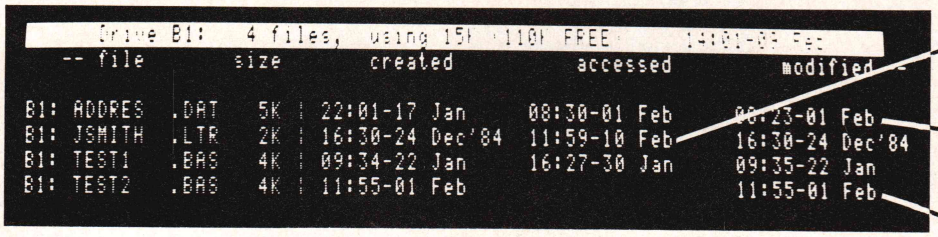
Peephole Optimizer

(Listing continued, text begins on page 56)

Listing Five

```
        if (locate(*lpos, &(pat[1])))
                advance = 1;
        break;
case NCCL:
        if (*lpos != NEWLINE && ! locate(*lpos, &(pat[1])))
                advance = 1;
        break;
default:
        message(pat);
        error("in omatch: can't happen");
}
if (advance >= 0)
{       *linp += advance;
        return(TRUE);
}
else
        return(FALSE);
}

/* patsize -- Returns size of pattern entry at pat. */

patsize(pat)
char *pat;
{
switch (*pat)
{
case LITCHAR:
case GRPPAT:
        return(2);
case BOL:
case EOL:
case ANY:
case GRPST:
case GRPEND:
        return(1);
case CCL:
case NCCL:
        return(pat[1] + 2);
case CLOSURE:
        return(CLOSIZE);
default:
        message(pat);
        error("in patsize: can't happen");
}
}

/* locate -- Look for c in character class at pat. */

locate(c, pat)
char c, *pat;
{
/* int i;
/*
/* /* Size of class is at pat, characters follow. */
/* for(i = *pat; i > 0; --i)
/* {    if (c == pat[i])
/*      {       return(TRUE);
/*      }
/* }
/* return(FALSE);
 C code commented out. */
#asm
; Assembly language version for speed
        POP     B       ;return address
        POP     H       ;pat
        POP     D       ;c
        PUSH    D       ;restore
        PUSH    H       ; the
        PUSH    B       ;  stack
        MOV     C,M     ;get count (*pat)
        MVI     B,0
        MOV     A,E     ;c into A
        INX     H       ;point to first character
#ifdef Z80
        DB      0EDH,0B1H       ;Z80 CPIR instruction
        JZ      locfnd  ;jump if c found
#else
loclp:  MOV     E,M     ;get char
```


DATESTAMPER™ has the answers



When did we print that letter?

Has the mailing list been updated?

Which is the latest version?

DateStamper™ keeps your CP/M computer up-to-date!

- avoid erasing the wrong file
- back-up files by date and time
- keep dated tax log of computer use
- simplify disk housekeeping chores

OPERATION: DateStamper extends CP/M 2.2 to automatically record date and time a file is created, read or modified. DateStamper reads the exact time from the real-time clock, if you have one; otherwise, it records the order in which you use files. Disks prepared for datestamping are fully compatible with standard CP/M.
INSTALLATION: Default (relative-clock) mode is automatic. Configurable for any real-time clock, with pre-assembled code supplied for all popular models. Loads automatically at power-on.
UTILITIES: Enhanced SuperDirectory • Powerful, all-function DATSWEEP file-management program with date and time tagging • Installation and configuration utilities
PERFORMANCE: Automatic. Efficient. Versatile. Compatible.

Requires CP/M 2.2. Uses less than 1K memory. Real-time clock is optional.

When ordering please specify format

8" SSSD, Kaypro, Osborne Formats **$49**
For other formats (sorry, no 96 TPI) add $5.
Shipping and handling **$3**
California residents add **6%** sales tax

MasterCard and Visa accepted

Specialized versions of this and other software available for the Kaypro.
CP/M is a registered trademark of Digital Research, Inc.

Write or call for further information

Plu*Perfect Systems

BOX 1494 • IDYLLWILD, CA 92349 • 714-659-4432

Circle **no. 69** on reader service card.

```
        CMP     E       ;compare with c
        JZ      locfnd  ;jump if match
        INX     H       ;increment pointer
        DCX     B       ;decrement count
        MOV     D,A     ;save A
        MOV     A,B
        ORA     C       ;test for BC == 0
        MOV     A,D     ;restore A
        JNZ     loclp   ;loop if BC != 0
#endif
        LXI     H,FALSE ;return FALSE if not found
        JMP     locend  ;exit from end for profiler
locfnd: LXI     H,TRUE  ;return TRUE if found
locend: DS      0
#endasm
}

/* makepat -- Make pattern from arg, terminate at delim. */

char *
makepat(arg, delim, stpat, ppat)
char delim, *arg, *stpat, **ppat;
{
char  *lp, *starg;
char *lastp, *tmp, *esc();

starg = arg;
lastp = *ppat;
while (*arg != delim && *arg != ENDSTR)
{       lp = *ppat;
        if (*arg == ANY)
                addpat(ANY,stpat,ppat);

        else if (*arg == BOL && arg == starg)
                addpat(BOL,stpat,ppat);
        else if (*arg == EOL)
                addpat(EOL, stpat,ppat);
        else if (*arg == CCL)
        {       if (! getccl(&arg, ppat, stpat))
                        return(NULL);
        }
        else if (*arg == CLOSURE && arg > starg)
        {       if (*(lp = lastp) == BOL || *lp == EOL || *lp == CLOSURE)
                        return(NULL);   /* Closure on a null string
                                           not allowed. */
                stclose(ppat,lastp,stpat);
        }
        else if (*arg == ESCAPE && arg[1] == '(')
        {       addpat(GRPST, stpat, ppat);
                arg += 2;
                if (++lparen > 9) return(NULL);
                if ((arg = makepat(arg, delim, pat, ppat)) != NULL)
                        --arg;
                else
                        return(NULL);
        }
        else if (*arg == ESCAPE && arg[1] == ')')
        {       addpat(GRPEND, stpat, ppat);
                ++rparen;
                return(arg+2);
        }
        else if (*arg == ESCAPE && ('0' < arg[1] && arg[1] <= '9'))
        {       addpat(GRPPAT, stpat, ppat);
                addpat(arg[1], stpat, ppat);
                ++arg;
        }
        else
        {       addpat(LITCHAR, stpat, ppat);
                tmp = esc(arg);
                addpat(*arg, stpat, ppat);
                arg = tmp - 1;
        }
        lastp = lp;
        arg++;
}
if (*arg != delim)
        return(NULL);
if(! addpat(ENDSTR, stpat, ppat))
        return(NULL);
if(lparen != rparen)
        return(NULL);
return(arg);
}

addpat(c, stpat, ppat)          /* Add c at ppat, check for overflow. */
char c, **ppat, *stpat;
{
if (*ppat - stpat > MAXPAT)

        return(FALSE);
*(*ppat)++ = c;
return(TRUE);
}

/* getccl -- Expand char class at arg into pat. */

getccl(argp, patp, stpat)
char **argp, **patp, *stpat;
{
char *ccst;
(*argp)++;      /* skip [ */
if (**argp == NEGATE)
{       addpat(NCCL, stpat, patp);
        (*argp)++;
}
else
        addpat(CCL, stpat, patp);
ccst = *patp;
addpat(0,stpat, patp);  /* space for count */
dodash(CCLEND, argp, stpat, patp);
*ccst = *patp - ccst - 1;
return(**argp == CCLEND);
}

/* stclose -- Insert closure entry at *patp. */

stclose(patp, lastp, stpat)
char **patp, *lastp, *stpat;
{
char *pp, *q;

for (pp = *patp - 1; pp >= lastp; pp--)
{       q = pp + CLOSIZE;
        addpat(*pp, stpat, &q);
}
*patp += CLOSIZE;
*lastp = CLOSURE;
}

/* esc -- Map *str into escaped character, return pointer to
          next character. */
```

(Continued on next page)

THE HAMMER
Software Tools in C

More than just BIOS/DOS access, THE HAMMER Library also provides routines for multi-level **123-like menus**, easy **data entry & verification** of strings, numbers, & dates, screen attribute control, date & time processing, AND MORE.

Super data entry routines give programmer a natural, strong interface with the user. They work in both "single-field" and "multi-field" (**full screen editing**) modes. This is NOT just another library of general purpose routines.

For: IBM/PC family with DOS 2.00+
C Compilers: CI-C86, Lattice, DeSmet, and new Microsoft V3.00

$195 with source code and manual. To order or inquire, **CALL OR WRITE:**

O.E.S. SYSTEMS
1906 Brushcliff Road
Pittsburgh, PA 15221
412/243-7365

Looking for the right tool for the job?
GET THE HAMMER

Circle **no. 87** on reader service card.

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)

OUTSIDE OKLAHOMA: NO SALES TAX

8087-3	MATH		$105.00
8087-2	COPROCESSORS		140.00
DYNAMIC RAM			
256K	256Kx1	120 ns	$ 4.49
256K	256Kx1	150 ns	3.25
64K	64Kx1	150 ns	.99
EPROM			
27C256	32Kx8	250 ns	$15.99
27256	32Kx8	250 ns	9.10
27128	16Kx8	250 ns	3.47
27C64	8Kx8	200 ns	7.85
2764	8Kx8	250 ns	2.50
2732A	4Kx8	250 ns	2.75
STATIC RAM			
6264LP-15	8Kx8	150 ns	$4.99
6116LP-3	2Kx8	150 ns	1.95

ADD 640 Kbyte TO PC/XT 8-SLOT MOTHERBOARD CONSUME NO EXPANSION SLOTS: $78.50

QUANTITY ONE PRICES SHOWN

OPEN 6½ DAYS: WE CAN SHIP VIA FED-EX ON SAT.

MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts µP∞
MICROPROCESSORS UNLIMITED
24,000 S. Peoria Ave., BEGGS, OK. 74421 **(918) 267-4961**

NO EXTRA COST ON F-EX SAT DELIVERY

Prices shown above are for July 2, 1985
Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 5 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ $6.00, or Priority One @ $15.00!

Circle **no. 64** on reader service card.

HISOFT
HIGH QUALITY SOFTWARE CP/M

HiSoft has been selling Z80 CP/M software in Britain and Europe for over 4 years. Now we'd like to introduce **you** to our range of programming languages:

HiSoft Devpac: Z80 assembler/editor/debugger
HiSoft C: Kernighan/Ritchie implementation
HiSoft Pascal: fast, **standard** compiler

All at $69 inclusive each.

These programs are also available for other Z80 machines including Timex 2068.
Call or write for full technical details and press commentaries, or order from:

HISOFT
180 High St. North
Dunstable LU6 1AT
ENGLAND
01144 (582) 696421

Circle **no. 48** on reader service card.

Peephole Optimizer (Listing continued, text begins on page 56)

Listing Five

```
char *
esc(str)
char *str;
{
int n, c;
char *p;
char *strcpy();

if (*str != ESCAPE)
        return(++str);
else if (*(p=(str+1)) == ENDSTR)        /* ESCAPE not special at end. */
        return(++str);
else
{       if (*p == 'N' || *p == 'n')
        {       *str = '\n';
                return(strcpy(++str,++p));
        }
        else if (*p == 'T' || *p == 't')
        {       *str = '\t';
                return(strcpy(++str,++p));
        }
        else if (*p == 'B' || *p == 'b')
        {       *str = '\b';
                return(strcpy(++str,++p));
        }
        else if ((*p | 0x20) == 'r')
        {       *str = '\r';
                return(strcpy(++str,++p));
        }
        else if (*p == '0' && (*(p+1) | 0x20) == 'x')   /* Hex */
        {       p += 2;
                *str = 0xff & xtoi(p);
                return(strcpy(++str,p += 2));
        }
        else if (*p >= '0' && *p <= '7')        /* Octal representation */
        {       for (n=0,c=0; (*p >= '0' && *p <= '7') && n<=3; ++n,++p)
                {       c = (c<<3) + (*p - '0');
                }
                *str = c;
                return(strcpy(++str,p));
        }
        else
        {       *str = *p;
                return(++p);
        }
}
}

/* dodash -- Expand set at src[i] into dest[j],
        stop at delim. */

dodash(delim, srcp, stdest, destp,)
char **srcp, **destp, *stdest, delim;
{
char *src, *cp, *tmp;
int k, junk;

src = cp = *srcp;
while (*src != delim && *src != ENDSTR)
{       if (*src == ESCAPE)
        {       tmp = esc(src);
                addpat(*src, stdest, destp);
                src = tmp-1;
        }
        else if (*src != DASH)
                addpat(*src, stdest, destp);
        else if ( src == cp || src[1] == ENDSTR)
                addpat(DASH, stdest, destp);    /* literal - */
        else if (isalnum(*(src-1)) && isalnum(src[1])
                && *(src-1) <= src[1])
                {       for(k = *(src-1) + 1; k <= src[1]; ++k)
                                addpat(k, stdest, destp);
                src++;
                }
        else
                addpat(DASH, stdest, destp);
        src++;
}
*srcp -= *srcp - src;
}

/* makesub -- Make substitution string from arg in sub. */

char *
makesub(arg, delim, sub)
char *arg, delim, *sub;
{
char *p, *q, *stsub, *tmp;

stsub = sub;
while (*arg != delim && *arg != ENDSTR && *arg != '\n')
{       if (*arg == '&')
        {       addpat(DITTO, stsub, &sub);
                addpat('0', stsub, &sub);
        }
        else if (*arg == ESCAPE && arg[1] > '0' && arg[1] <= '9')
        {       addpat(DITTO, stsub, &sub);
                addpat(arg[1], stsub, &sub);
                ++arg;
        }
        else
        {       tmp = esc(arg);
                addpat(*arg, stsub, &sub);
                arg = tmp - 1;
        }
        ++arg;
}
if (*arg != delim)
        return(NULL);   /* Missing delimiter */
if (! addpat('\0', stsub, &sub))
        return(NULL);   /* No space for sub string */
else
        return(arg);
}

/* subst -- Substitute "sub" for occurrences of pattern. */
/* Modified for optimizer. */

subst(ln, tr)
int ln;
struct trfmn *tr;
{


struct lkline *oo, *nn;
char *newline();
int line, n;

lparen = rparen = 0;
nn = tr->new;
oo = tr->old;
for (n = tr->nold; n > 0; --n)
{       if ((line = ln - n + 1) < 0)
                line += maxlines;       /* Buffer wrap around. */
        if (n <= tr->nold - tr->nnew) {
                delln(line);    /* Not enough replacements. */
                oo = oo->nextln;
        }
        else {
                if (!subline(line, oo->ln, nn->ln))
                        return(FALSE);
                else
                {       oo = oo->nextln;
                        nn = nn->nextln;
                }
        }
}
for (n = tr->nnew; n > tr->nold; --n) {         /* More lines to add. */
        newline("\n");
        if (!subline(curln, BOLSTR, nn->ln))
                return(FALSE);
        else
                nn = nn->nextln;
}
return(TRUE);
}

/* subline -- Do substitution on one line. */
subline(line, pat, sub)
int line;
char *pat, *sub;
{
char new[MAXSTR];
char *p1, *p2, *p3, *last, *pp;
int subbed;
char *amatch(), *strcpy(), *malloc();

p2 = new;
subbed = FALSE;
last = NULL;
p1 = linep[line];
while (*p1 != ENDSTR)
{       if (!subbed)
        {       pp = pat;
                p3 = amatch(linep[line], p1, &pp);
        }
        else
                p3 = NULL;
        if (p3 != NULL && last != p3)


        {       subbed = TRUE;          /* replace matched text */
                catsub( p1, p3, sub, new, &p2);
                last = p3;
        }
        if (p3 == NULL || p3 == p1)
        {       addpat(*p1, new, &p2);
                ++p1;
        }
        else    /* Skip matched text. */
                p1 = p3;
}
if (subbed)
{       if (!addpat(ENDSTR, new, &p2))
        {       return(FALSE);
```

(Continued on page 82)

BRIEF™

"BRIEF Is The Best Editor I Have Ever Used. I Switched. I Use BRIEF Regularly."

Steve McMahon*

Choosing the right program editor is an extremely important decision. After all, more time is spent with your program editor than with any other category of software. A good program editor helps to stimulate creativity and produce programs of superior design. A poor program editor, on the other hand, can be worse than none at all. It can slow you down, get in your way, and make even the simplest tasks difficult.

BRIEF - PROGRAM EDITING *YOUR WAY*

Every programmer has an individual style that makes their work unique. Most program editors compromise that style by forcing the programmer to conform to their methods. Not so with BRIEF.

BRIEF's most powerful feature is its ability to conform to your way of programming. BRIEF can be easily tailored/customized to your individual preferences. For example, keyboard reassignment allows the keyboard to be used in whatever way you prefer. Keystroke macros allow long sequences of editor directives to be repeated by pressing a single key. Plus you can simultaneously work with numerous program and data files, and configure many windows to control BRIEF's visual presentation.

75% OF THE EXPERTS WHO HAVE TRIED BRIEF SWITCHED!

(Call For Details)

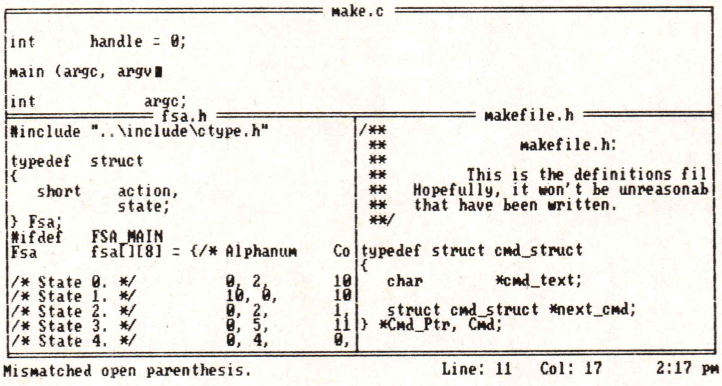
"A BONA FIDE UNDO" Steve McMahon – BYTE REVIEW "BUILD YOUR DREAM EDITOR" MARCH 1985

Here are Steve McMahon's exact words: "...BRIEF implements a true undo facility, by default allowing command-by-command recovery from the last 30 commands...The number of commands you want to be able to undo can be changed." (up to 300) "Only with BRIEF, though, was it possible to undo a macro that produced 4000 words of text with a single keystroke."

EVERY FEATURE YOU WANT

BRIEF supports practically every feature you've ever seen, and some you probably haven't thought of yet.

- Edit Multiple Large Files
- Windows (Tiled & Pop-up)
- Unlimited File Size
- Full UNDO (N Times)
- True Automatic Indent for C
- Exit to DOS Inside BRIEF
- Compile programs inside BRIEF
- Uses All Available Memory
- Intuitive Commands
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning
- Reconfigurable Keyboard
- Online Help
- Search for "Regular Expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- And ... a Complete, Compiled, Programmable and Readable Macro Language.



A TYPICAL BRIEF SCREEN

Notice there are three windows on the screen simultaneously and each one is viewing a different file. The mainline of a C program is visible in the uppermost window; the programmer has run a syntax checking macro which found a mismatched open parenthesis in the arguments to the mainline. The other two windows show header files containing information crucial to the design of the program. BRIEF can have an unlimited number of windows and files accessed simultaneously.

BRIEF'S ONLY LIMITATIONS ARE THE ONES *YOU* SET

Full refund if not satisfied in 30 days. ONLY $195

→ **FREE** with order: "Best of BRIEF Macros" — Contest entries include macros for Fortran, C, Calculator, Base Converter, etc. Call before November 30.

AVAILABILITY: IBM-PC and Compatibles, AT, & TANDY 2000

Solution Systems™

335-D Washington St., Norwell, MA 02061 (617) 659-1571

*Steve McMahon's quote courtesy Suntype Publishing Systems. BYTE review by Mr. McMahon may be found in BYTE Magazine March 1985.

Circle **no. 102** on reader service card.

Peephole Optimizer

(Listing continued, text begins on page 56)

Listing Five

```
        }
        else {  /* Copy new line into linep array. */
                free(linep[line]);
                if ((linep[line] = malloc(strlen(new)+1)) == NULL)
                        error("not enough memory");
                strcpy(linep[line], new);
        }
}
return(subbed);
}

/* catsub -- Add replacement text to end of new. */

catsub(p1, p2, sub, new, pp3)
char *p1, *p2, *sub, *new, **pp3;
{
char *p, *q, *esc();

for (p = sub; *p != ENDSTR; ++p)
{       if (*p == DITTO)
        {
                if (p[1] > '0' && p[1] <= '9')
                {       p1 = qstart[p[1]-'0'];
                        p2 = qend[p[1]-'0'];
                }
                for (q = p1; q < p2; ++q)
                {       addpat(*q, new, pp3);
                }
                ++p;
        }
        else
        {       addpat(*p, new, pp3);
        }
}
}

/* delln -- Delete a line. */

delln(ln)
int ln;
{
free(linep[ln]);
linep[ln] = NULL;
}
```

End Listing Five

Listing Six

```
SHLD\t\([\._a-zA-Z_][\._a-zA-Z0-9+\-]*\)
^\tLHLD\t\1
%
SHLD\t\1\t; redundant LHLD deleted
%%
\tMOV\tA,H
^\tORA\tL
\tJNZ\t\([\._a-zA-Z][\._a-zA-Z0-9]*\)
^\tLXI\tH,0
%
\tMOV\tA,H
\tORA\tL
\tJNZ\t\1\t; LXI H,0 removed following in line test
%%
\tCALL\t\([^en]*\)
\tJNZ\t\([\._a-zA-Z][\._a-zA-Z0-9]*\)
^\tLXI\tH,0
%
\tCALL\t\1
\tJNZ\t\2\t; LXI H,0 removed following library test
%%
\tCALL\t\([^en]*\)
\tJZ\t\([\._a-zA-Z][\._a-zA-Z0-9]*\)
^\tLXI\tH,0
%
\tCALL\t\1
\tJZ\t\2\t; LXI H,0 replaced following library test
\tDCX\tH
```


TAKE THE PAIN OUT OF BACKUP

Famous Excuses For Not Doing Backups

Backing up a Winchester onto multiple floppies takes forever.

PIP makes you remember what you changed — too much trouble.

Disk failures only happen to the other guy.

Floppy-To-Floppy Incremental Backup For Only $40.

Like PIP but copies just the files that have changed.

Order Qbax1.

The Remedy: Copy Just What's New — Automatically!

Qbax2 will:
Split files bigger than one floppy.
Track all the pieces.
Tell you which floppy it needs and when.
Give you a built-in catalogue.
Reclaim wasted floppy space.

Qbax2 only $95 from:

Amanuensis, Inc.
R. D. 1, Box 236
Grindstone, PA 15442
(412) 785-2806

For CP/M 2.2 on 8" SSSD and popular 5¼" formats. Shipping: $2 U.S. and Canada, $4 overseas. Qbax TM Amanuensis, Inc. CP/M® Digital Research.

FOR EVERYONE WITHOUT A DISPENSATION FROM MURPHY'S LAW

Circle **no. 3** on reader service card.

```
%%
\tCALL\t\([cen]\.\)
\tJZ\t\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\tLXI\tH,1$
%
\tCALL\t\1
\tJZ\t\2\t; LXI H,1 replaced following library test
%%
^\tJZ\t\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\tJMP\t\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\1\(:\t.*\)
%
\tJNZ\t\2\t; JZ around jump removed
\1\3
%%
^\tJNZ\t\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\tJMP\t\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\1\(:\t.*\)
%
\tJZ\t\2\t; JNZ around jump removed
\1\3
%%
PUSH\tH
^\tLXI\tH,\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\tPOP\tD

%
XCHG\t;PUSH H POP D  around LXI H changed to XCHG
\tLXI\tH,\1
%%
PUSH\tH
^\tLHLD\t\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\tPOP\tD
%
XCHG\t;PUSH H POP D  around LHLD changed to XCHG
\tLHLD\t\1
%%
STA\t\([\.a-zA-Z_]\)[\._a-zA-Z0-9+\-]*\)
^\tLDA\t\1
%
STA\t\1\t; redundant LDA deleted
%%
\tJMP\t
^\(^\t[^D][^BSW]\)
%
\tJMP\t
; unreachable code:   \1
%%
\tRET$
^\t
%
\tRET
\t; unreachable code:
%%
\tJMP\t\([\._a-zA-Z]\)[\._a-zA-Z0-9]*\)
^\1\(:\t.*\)
%
\tDS\t0\t; jump to next location removed
\1\2
%%
```

End Listing Six

Listing Seven

```
/* Library routines used by optimizer. */

atoi(str)                       /* Convert str to integer. */
     char *str;
exit()                          /* Return to CP/M */
fclose(fd)                      /* Close file with descriptor fd. */
     FILE fd;
char *fgets(str, nbytes, fd)    /* Read one line (maximum of nbytes) */
     char *str;                 /* from file into str. */
     int nbytes;
     FILE fd;
FILE fopen(name, mode)          /* Open file name with mode="r" or "w". */
     char *name, *mode;
fprintf(fd, format [, arg ] ... )  /* Formatted output routine. */
     FILE fd;
     char *format;
fputs(str, fd)                  /* Output str to file fd. */
     char *str;
     FILE fd;
free(block)                     /* Release block of memory allocated by
     char *block;               /* malloc. */
isalnum(c)                      /* Returns non-zero (true) if c is a */
     char c;                    /* letter or digit. */
char *malloc(nbytes)            /* Return pointer to block of nbytes */
     int nbytes;                /* of available memory. */
putc(c, fd)                     /* Output character c to file fd. */
     char c;
     FILE fd;
char *sbrk(nbytes)              /* Return pointer to nbytes of memory. */
     int nbytes;                /* Cannot be freed. */
char *strchr(str, c)            /* Return pointer to first occurrence of c */
     char *str, c;              /* in str. */
strcmp(str1, str2)              /* Compare strings 1 and 2. Return 0 if */
     char *str1, *str2;         /* str1==str2, negative if str1<str2, or */
                                /* positive if str1>str2. */
char *strcpy(dest, source)      /* Copy string at source to dest. */
     char *dest, *source;
strlen(str)                     /* Return length of str. */
     char *str;
strncmp(str1, str2, n)          /* Compare first n characters of str1 and */
     char *str1, *str2;         /* str2.  Return 0 if identical, negative */
     int n;                     /* if str1<str2 or positive if str1>str2. */
xtoi(str)                       /* Return integer value of hexadecimal */
     char *str;                 /* representation in str. */
```

End Listings

Finally. BSW-Make.

The Boston Software Works now brings a complete implementation of the Unix "make" facility to MS-DOS. No more recompiling every file in sight after a small edit; no more wondering if you've really rebuilt every module affected by an edit. Just type "make" and BSW-Make automatically builds your product quickly, efficiently and correctly.

BSW-Make supports:

- most compilers and assemblers
- MS-DOS or PC-DOS v2.00 or later
- macros for parameterized builds
- default rules
- MS-DOS pathnames
- any MS-DOS machine (192K minimum)

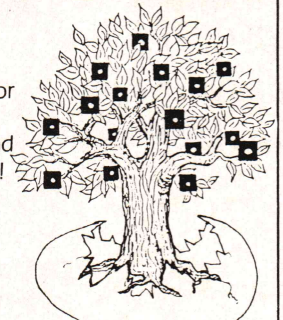
Only $69.95 postpaid (Mass. residents add 5% sales tax)

The Boston Software Works
120 Fulton Street, Boston, MA 02109
(617) 367-6846

Circle **no. 1** on reader service card.

Tree Shell

A Graphic Visual Shell for Unix/Xenix End-Users and Experts Alike!



Dealer inquiries welcomed.

COGITATE

"A Higher Form of Software"
24000 Telegraph Road
Southfield, MI 48034
(313) 352-2345
TELEX: 386581 COGITATE USA

Circle **no. 24** on reader service card.

THIS AD IS
ACTUAL PRINTED SIZE

MAKE YOUR MOUNTAINS INTO MOLEHILLS:
MINI-PRINT your PC-DOS Files

MINI-PRINT prints six full pages on each 8-1/2 x 11 sheet of printer paper, like this:

1	4
2	5
3	6

Over 38,000 chars/page
A 4:1 reduction in paper.

Bigger programs or masses of business data can be handled more easily.

Improves:
- Desk Space
- Archival Storage
- Paper Usage
- Price Lists
- Inventories
- Spread-Sheets

- Seeing the Big Picture
- Visualization
- Placement
- Comparison

MINI-PRINT uses the 216 dpi line-spacing + dual-density graphics of an Epson or similar IBM PC printer (MX-80, RX-80, FT-855, etc.), and quickly generates four lines with every three passes of the print-head, for a quantum leap in man - machine communications.

Use MINI-PRINT and you'll see the top of your desk again.

$25.00 (includes tax + shipping)
or call:

ZEBRA SYSTEMS
851 Haddock Street
Foster City, CA 94404
(415) 341-2011

Show this ad to your friends

Circle **no. 110** on reader service card.

Small-C Update

by J. E. Hendrix

What's the author of Small-C V2 been up to lately? Improvements, bug fixes, and a macro assembler.

It has been about a year since the last *Dr. Dobb's* article on the Small-C compiler, and many readers may be wondering about further developments. I have received a number of queries, mostly about structures and additional data types, but for several reasons I have not developed Small-C further.

For one thing, additions would affect nearly every part of the compiler, so it makes sense to clean up the overall structure of the compiler before proceeding. Rather than bloating the compiler with code, I would like to streamline its design, reduce its size, and improve its efficiency before adding more features. Then there is the need to free Small-C from dependence on other packages for assembling and linking its output. I consider this capability more important than additional compiler features. All of this amounts to a considerable project, and I just haven't had the time recently to spend on it.

I also wonder whether Small-C really needs to become a full C compiler to be a useful tool. It is already suitable for many applications; text processing and system utilities are obvious examples. With plenty of full-featured C compilers available at reasonable prices, I haven't felt a great urge to take Small-C beyond its present entry-level stature.

I have sent the compiler to individuals and companies involved in auto-ignition control; manufacturing of CRT terminals, electronic games, microcomputers, and circuit cards; mental health; publishing; materials management; telecommunications; engineering research; control of devices like transmitters and telescopes; and education. I have noticed an increasing interest in Small-C in education circles—high school, vocational-technical, and college. Because it is small, is written in a popular high-level language, and generates readable assembly language code, Small-C is ideal for studying what compilers do and how they work. Its one-pass algorithm and the fact that input and output default to the console make it especially convenient for studying the relationship between compiler input and output. The Small-C compiler is also a great guinea pig for term projects, giving hands-on experience in compiler construction to would-be computer scientists. And *The Small-C Handbook* makes it easy for users to quickly familiarize themselves with the language and the compiler.

I maintain a mailing list of Small-C users whom I try to keep informed of bug fixes and improvements. But because Small-C may be distributed freely (on a noncommercial basis), many people have copies that they obtained indirectly and that may be out of date. If you have version 2.1 with the new library that was introduced in May and June 1984 (*DDJ* No. 91 and No. 92), then the fixes reported below will bring you up to date.

I am still distributing Small-C for $25, *The Small-C Handbook* for $17.95, and a package of Small-C software tools for $35. Naturally, complete source code is included. To order, send a check or money order to me at the address listed at the end of the article. Include $4 for post-

---

J. E. Hendrix, Box 8378, University, MS 38677.

age and packaging plus $6 for overseas air mail. Send a self-addressed, stamped envelope for information.

People frequently ask for diskette formats other than 8-inch SSSD and North Star 5¼-inch, making it necessary for me to refer them elsewhere, and many people to whom I have sent Small-C and the tools have machines with different diskette formats, CPUs, and operating systems. In most cases, however, I do not know where they stand in porting the software or how willing they are to take referrals. So if you are running Small-C 2.1 (with the new library) on anything except an 8-inch or North Star CP/M machine and are willing to help distribute it, please let me know. Indicate (1) your machine, (2) your operating system, (3) the diskette formats you can supply, and (4) whether or not you have the Small-Tools package.

Redundant Code

David Bookbinder has observed that Small-C generates redundant code for deallocating local variables at the end of a function when the last statement is a return. The same is true if a return is the last statement in any compound statement that declares local variables. This is harmless, but it unnecessarily adds to the size of programs and it really is untidy.

Likewise, if the last statement in a function is a goto, unnecessary code for deallocating local variables and a RET is generated: because a goto would block the exit path from a function, there is no need for such code. (The compiler already eliminates the RET if the last statement is a return.) There is no problem with goto's terminating nested compound statements because goto's are not allowed if variables are declared anywhere after the opening brace of the function body—so there are no variables to deallocate.

The following patch eliminates this redundant code:

(1) Add the following line to the end of the file CC.DEF:

```
#define STLABEL 14
```

(2) Change the line in newfunc( ) (file CC12.C) that contains a call to statement( ) to read:

```
    statement( );
#ifdef STGOTO
    if(laststst != STRETURN && laststst != STGOTO)
        ffret( );
#else
    if(laststst != STRETURN) ffret( );
#endif
```


Sidekick for CP/M!

Poor Person Software brings you

Write-Hand Man

Desk Accessories for CP/M

Suspend CP/M applications such as WordStar, dBase, and SuperCalc, with a single keystroke and look up phone numbers, edit a notepad, make appointments, view files and directories, communicate with other computers. Return to undisturbed application! All made possible by **Write-Hand-Man.** Ready to run after a simple terminal configuration! No installation.

Don't be put down by 16 bit computer owners. Now any CP/M 2.2 machine can have the power of *Sidekick.*

Bonus! User extendable! Add your own applications to **Write-Hand-Man.** All you need is M80 or RMAC.

$49.95 plus tax (California residents), shipping included! Volume discounts.

Available on IBM 8 inch and Northstar 5 inch disks. Other 5 inch formats available with a $5.00 handling charge. CP/M 2.2 required.

COD or checks ok, no credit cards or invoices

Poor Person Software

3721 Starr King Circle

Palo Alto, CA 94306

tel 415-493-3735

Write-Hand-Man trademark of **Poor Person Software,** CP/M and RMAC trademarks of Digital Research, *Sidekick* trademark of Borland International, dBase trademark of Ashton-Tate, WordStar trademark of Micropro, SuperCalc a trademark of Sorcim, M80 trademark of Microsoft.

Circle **no. 71** on reader service card.

WIZARD C

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language

February, 1985

- All UNIX System V language features
- Support for 8087, 80186 and 80286
- Full library source code included
- Cross-file checks (full UNIX lint)
- Uses MS-LINK or PLINK 86
- ROMable data options
- In-line assembly language
- Cross compilers available
- Third party software available, including PANEL

The new standard for C Compilers on MSDOS!

Only $450.

(617) 641-2379

WIZARD Systems Software, Inc.

11 Willow Court

Arlington, MA 02174

VISA MasterCard

Circle **no. 116** on reader service card.

(3) Change the line in statement( ) (file CC13.C) that contains a call to dolabel( ) to read:

```
else if(dolabel( )) laststst=STLABEL;
```

(4) Change the line in compound( ) (file CC13.C) that contains a call to modstk( ) to read:

```
#ifdef STGOTO
   if(laststst != STRETURN && laststst != STGOTO)
#else
   if(laststst != STRETURN)
#endif
      modstk(savcsp, NO); /* delete local variable
         space */
   csp=savcsp;
```

(5) Recompile, assemble, and link the compiler.

FPUTC Placement

A problem has been reported concerning the placement of the module FPUTC in the relocatable library CLIB.REL. Its placement between FPUTS and FREAD makes it precede PUTCHAR, which makes reference to it. Because L80 can't load a module that it has already passed, programs (like "words" on page 31 of *The Small-C Handbook*) that reference putchar( ) but not fputc( ) will not link properly. The correction for this condition is to move FPUTC down in CLIB.REL so that it falls between FOPEN and FREE. This is done by invoking LIB80 as follows:

```
A>LIB80
*NEWLIB=
*CLIB<. . FPUTS>
*CLIB<FREAD . . FOPEN>
*CLIB<FPUTC>
*CLIB<FREE . . >
*/E
```

You should also change the placement of FPUTC in the file NEWLIB3.SUB so it will order the modules correctly if you ever build a new library from scratch. As a precaution, you should have LIB80 list the contents of the new library (see notes below) then verify that the order exactly matches NEWLIB3.SUB. Finally, delete CLIB.REL and rename NEWLIB.REL to CLIB.REL.

fflush( ) and Auxiliary Buffers

The function fflush( ) attempts to flush an auxiliary buffer even if the file were opened for reading only. In this case, the auxiliary flush routine returns an error condition when a normal return without taking any action would

EC Text Editor

. . . the Latest in Programming Environment Technology . . .
The DOS Interface — All Available Memory — Windows !!!

EC's DOS Interface is a major advance. It is far more sophisticated and useful than the "EXIT" to DOS provided by other editors.

Here's how you can use the DOS Interface:
1) Execute any DOS command you like.
2) Keep a history of all DOS output (you can even have EC automatically save it to a file if you want to keep a DOS Log.) EC displays the output in the DOS Interface. You can page through it, scroll it, etc.
3) You can re-execute or edit previous commands, even those very far back in the DOS history.
4) You can set a prompt, just like you would for DOS.

Here's how this can help you: You can run a database manager or a compiler inside the editor. Enter the command to execute your program just like you would in DOS. If you get error messages, exit the application, open a window on the error messages, and view the error messages as you edit your text.
Jump instantly to the lines with errors. Correct all of the syntax errors after one compile; it's faster and more convenient than getting thrown back in your text time and time again as the compiler finds each error.

The DOS Interface captures ALL DOS output — even from your application. When your program terminates, and you want to edit your source code, you can still view your program's previous output! Especially useful as a debugging trace.

Read in large files — limited by your PC's memory. Move text between windows and buffers.

There's more: List command — like a random access Find, wildcards for Find/Replace/List, tab size dependent on file extension, auto-indent for C, on-screen file-comparison, Path support, support for read-only files, extensive screen and printer support, International and other extended ASCII characters, support for International keyboards, garbage pile of deleted text, columnar editing. Keystroke macros are created merely by hitting the keys — no codes to remember.

On-line tables: ASCII, extended ASCII, extended keyboard codes, operator precedence for C, DOS and BIOS functions, PC Memory Map. **On-line Help** and a **Tutorial.** EC was written with the DeSmet C compiler on a PC.

Please note: EC's low price belies its quality. EC stands with the best! The price is low because we want everyone to try EC and find out why it is superior to editors costing $200 and more!

EC is not copy-protected. **30 day money back guarantee.** Full-featured demo: $5.00.

EC Editor	**$ 49.50**
DeSmet C Compiler	**99.50**
DeSmet Debugger	**50.00**
BASIC_C Library	**175.00**
Lattice C Compiler	**350.00**
PC-Lint	**89.50**

EC and BASIC_C run on PC compatibles.
We use and support all products we sell.

C Source
12801 Frost Road
Kansas City, MO 64138
(816) 353-8808

Shipping: $5.00 for orders under $50.00. MC, VISA, COD. Site licenses available. OEM's, dealers welcome.

Other Products

The DeSmet C Compiler — the fastest professional C compiler for the PC! DeSmet's program speed & efficiency are excellent. We chose DeSmet to develop EC and Basic_C, though, because of one factor: compile and link time. DESMET IS FOUR TIMES FASTER THAN LATTICE. 'Nuff said.

The Basic_C Library — the Convenience and Rich Function Set of BASIC for C.
The best way to move BASIC code over to C. The manual is regarded as the best tutorial for programmers moving from BASIC to C. You get a version for Lattice, DeSmet, and CI. No royalties. Includes source.

PC-Lint — A top quality lint checker at a quarter the price of the competition. Save hours of debugging by catching subtle bugs like function return mismatches.

Circle **no. 19** on reader service card.

have been appropriate. Before auxiliary buffering was added, it was not necessary to check the open mode because the "dirty buffer" status was being checked and it could not have been set for read-only files. This fix verifies the open mode before any attempt to flush a buffer. An attempt to flush a closed file now gives a normal return. Modify fflush( ) as shown below, compile, assemble, and replace in CLIB.REL:

```
FFLUSH.C
  ...
  fflush(fd) int fd; {
    if(Umode(fd) & WRTBIT) {
      if((Uauxsz && Uauxsz[fd] && Uauxfl(fd)) ||
          (!isatty(fd) && Udirty[fd] &&
           Usector(fd, WRTRND))) {
        Useterr(fd);
        return (ERR);
        }
    }
  return (NULL);
  }
```

Reset Auxiliary Buffer Pointers

When an fd that has an auxiliary buffer (because of a previous call to auxbuf( )) is closed and reopened, the next-byte and end-of-data auxiliary buffer pointers are not reset. This causes improper reading and writing. You must revise two library modules to correct this problem: CSYSLIB and AUXBUF. These modules should be revised as indicated below, compiled, assembled, and replaced in CLIB.REL:

```
CSYSLIB.C
  ...
  int
    *Uauxsz,   /* addr of Uxsize[ ] in AUXBUF */
     Uauxin,   /* addr of Uxinit( ) in AUXBUF */
     Uauxrd,   /* addr of Uxread( ) in AUXBUF */
     Uauxwt,   /* addr of Uxwrite( ) in AUXBUF */
     Uauxfl,   /* addr of Uxflush( ) in AUXBUF */
  ...
  Uopen(fn, mode, fd) char *fn, *mode; int fd; {
    char *fcb;
    if(!strchr("rwa", *mode)) return (ERR);
    Unextc[fd] = EOF;
    if(Uauxin) Uauxin(fd);
    if(strcmp(fn, "CON:")==0) {
      Udevice[fd]=CPMCON; Ustatus[fd]=
        RDBIT | WRTBIT; return (fd);
      }
  ...
```

Circle **no. 84** on reader service card.

ConIX™

NOW ONLY $79.95!

If you think you're missing out on innovative software developments because nobody is writing for CP/M™-80, take a look at us. We've adapted UNIX™ features to CP/M like never before, and with the kind of professional, quality-controlled product that you deserve. That product is none other than the critically acclaimed ConIX Operating System.

ConIX can provide any 48K+ CP/M-80 or compatible system with I/O Redirection and Pipes (uses memory or disk), perfected User Areas, Command and Overlay Path Searching, Auto Screen Paging, 8Mb Print Buffering, 22 new SysCalls, Function Keys, "Virtual" disk system, Archiver (saves over 50% disk), extensive command language, 300+ variables, 100+ commands, pull-down menu, and much more! Uses as little as 1/2K RAM! Runs with CP/M for true data and software compatibility. Installs easily without any system mods!

The ConIX package lists at $165 and has been advertised and sold internationally to many enthusiastic customers since October 1983. As a special limited offer, we've lowered the price of the complete ConIX system by 50% to only $79.95! Don't miss this opportunity to bring your 8-bit micro back into the software revolution. Order your copy of ConIX today!

Price includes manual, 8" disk, and user support. 5¼" conversions available. Contact your local dealer, or buy direct and add shipping: $4.50 UPS, $10 Canada, $25 overseas. NY residents add sales tax.

Computer Helper Industries Inc.
P.O. Box 680 Parkchester Station, NY 10462
Tel. (212) 652-1786 (for information/orders)

"We're helping your computer work better for you!"

UNIX: AT&T Bell Labs, CP/M: Digital Research, ConIX: Computer Helper Ind.

Circle **no. 22** on reader service card.

AUXBUF.C

```
...
extern int *Uauxsz, Uauxin, Uauxrd, Uauxwt,
    Uauxfl, Ustatus[ ];
...
auxbuf(fd, size) int fd; char *size; { /* fake
    unsigned */
    if(!Umode(fd) || !size || avail (NO) <
        size || Uxsize[fd])
      return (ERR);
    Uxaddr[fd] = malloc(size); Uxinit(fd);
    Uauxin = Uxinit;
        /* tell Uopen( ) where Uxinit( ) is */
    Uauxrd = Uxread;
        /* tell Uread( ) where Uxread( ) is */
    Uauxwt = Uxwrite;
        /* tell Uwrite( ) where Uxwrite( ) is */
    Uauxsz = Uxsize;
        /* tell both where Uxsize[ ] is */
    Uauxfl = Uxflush
        /* tell fflush( ) where Uxflush( ) is */
    Uxsize[fd] = size;
        /* tell Uread( ) that fd has aux buf */
    return (NULL);
    }
/*
** Initialize aux buffer controls
*/
Uxinit(fd) int fd; {
    Uxnext[fd] = Uxend[fd] = Uxaddr[fd];
    Uxeof[fd] = NO;
    }
...
```

Optimizing Tests Against Zero

Small-C optimizes tests against the value zero. However, in its enthusiasm, it overlooks certain unary operators that might spoil its efforts. So it optimizes if(!(i==0)) . . . ; as though it were if(i==0) . . . ;. You can fix this by making the changes indicated below to file CC32.C of the compiler:

CC32.C

```
...
hier13(lval) int lval[ ]; {
...
    else if (match("~")) {      /* ~ */
        ...
        return (lval[7]=0);
        }
    else if (match("!")) {      /* ! */
        ...
        return (lval[7]=0);
        }
    else if (match(" - ")) {    /* unary - */
        ...
        return (lval[7]=0);
        }
    ...
```

Lexcmp( )

Lexcmp( ) erroneously returns zero, indicating a match, when the first bytes that are not identical are upper and lower cases of the same letter; for instance, "Happy" and "HAY" appear to match. Revise lexcmp( ) as indicated below, recompile, assemble, and replace in CLIB.REL:

LEXCMP.C

```
...
lexcmp(s, t) char *s, *t; {
    while(lexorder(*s, *t) == 0)
        if(*s++) ++t;
        else return (0);
    return (lexorder(*s, *t));
    }
...
char Ulex[128] = {
            /**** NUL '- / ****/
              0,  1,  2,  3,  4,  5,  6,  7,  8,
              9, 10, 11, 12, 13, 14, 15, 16, 17,
             18, 19, 20, 21, 22, 23, 24, 25, 26,
             27, 28, 29, 30, 31, 32, 33, 34, 35,
             36, 37, 38, 39, 40, 41, 42, 43, 44,
             45, 46, 47,
            /**** 0-9 ****/
             65, 66, 67, 68, 69, 70, 71, 72, 73,
             74,
            /**** : ; < = > ? @ ****/
             48, 49, 50, 51, 52, 53, 54,
            /**** A-Z ****/
             75, 76, 77, 78, 79, 80, 81, 82, 83,
             84, 85, 86, 87, 88, 89, 90, 91, 92,
             93, 94, 95, 96, 97, 98, 99, 100,
            /**** [ \ ] ^ U ` ****/
             55, 56, 57, 58, 59, 60,
            /**** a-z ****/
             75, 76, 77, 78, 79, 80, 81, 82, 83,
             84, 85, 86, 87, 88, 89, 90, 91, 92,
             93, 94, 95, 96, 97, 98, 99, 100,
            /**** { | } ~ ****/
             61, 62, 63, 64,
            /**** DEL ****/
             101
            };
...
```

This patch also makes a minor change for improved efficiency and removes leading zeroes from the values in Ulex[ ] so they will not look like octal values to full C compilers.

Macro Storage

Experience has shown that the amount of storage space allocated for macro replacement strings is probably too small for the number of macros allowed. Also, the number of macros allowed may not be large enough for some programs.

The following parameters in CC.DEF have been increased to help this situation:

DDJ Classifieds

Software

NATIONAL PUBLIC DOMAIN

Why reinvent the wheel?
Public Domain Software isn't copyrighted, so no need to pay license fees for thousands of routines; business-utilities-dbase. Complete, latest issue user group disks available to rent 'n copy at your leisure. Send $5.00 for a postpaid directory disk or a SASE to National Public Domain Software, 1533 Avohill Dr., Vista, CA 92083 or (619) 727-1015.

FED "Binary File Editor"
Allows the user direct access in both HEX and ASCII format to any disk file on the IBM PC. FED can patch object modules, examine word processing files, repair damaged files and verify the results of I/O operations. S/S DOS disk for 128 IBM PC. Only $49 "+ $5 s/h".
The Whitewater Group
2912 N. Burling Avenue
Chicago, IL 60657
(312) 975-6095

Quality Software at Giveaway Prices!
We have to clear lots of CP/M & MSDOS software. Many 75% off list price. Source for many. Thousands of 5-inch & 8-inch diskettes—under $1 ea. "Everyman's Database Primer"/ "dBase II for Every Business"—$7.50 ea. + $2 S/H.
Software Salvage
Box 640
Norwalk, CT 06856

DBase II User—Converting to "C"
Try the dBx translation system— from DBase II to quality "C"; incl. translator, screen handler, & sort (w/source)—uses any file handler. For MSDOS, XENIX & UNIX Sys 5.
Desktop AI
Box 640
Norwalk, CT 06856

"COMPU' BLOKS"
4510 Holiday Blvd.
Salt Lake City, Utah 84117
(801) 272-2391
Compu' Bloks are a space odyssey for child or adult. Rectangles, Octagons, Squares and Circles create countless extensions in all directions. Space Stations, Lasars and Probes in an ordered sequence of infinite numbers . Comets, Meteors and Constellations. $.75 brings catalog. Rwon Lucasor

$50 FORM & SCREEN PAINTER FOR dBASE, BASIC & TURBO
100,000 copies shipped with dBASE II, now on PC & MSDOS. "Paint" prompts, data fields, lines, boxes where you want them and ZIP® writes dBASE II/III, BASIC or Turbo Pascal code. Then add one line to your programs for professional data entry and reports. $50 cash, check, or VISA, no COD.
MAGNUM DATA INC.
627 South Plymouth Blvd.
Los Angeles, CA 90005
(213) 937-0808

ECLIPSE SYSTEMS

AZTEC C65 + ProDOS
Use Aztec C DOS 3.3 software under ProDOS. Run programs without relinking. System includes recursive Shell, support for pseudo-code, vi like editor with macros, library upgrade and source code. Requires ProDOS user's disk. $49.95.
Eclipse Systems, 223 Matthew Rd., Marion, PA 19066. (215) 667-8354

W.B. SOFTWARE DEVELOPMENT

BROWSE for CP/M-68K
for looking at any CP/M-68K file
scroll up, down, left, right
powerful FIND (search) command
online HELP, PFKEYS, TABS, PRINT
100% 68000 assembler—ASCII, HEX
ANSI 3.64 & other terminal types
$49.95 US. Visa, M/C, money order.
WB Software Development,
112-Oakhampton Pl. SW,
Calgary, AB,
Canada T2V 4B2 (403) 238-3216

Utility

PRODUCE ASSEMBLY CODE FAST!
Basic XPL is a high-level assembler with source code portability between CP/M and PCDOS. Complete with detailed 65-page manual. Satisfaction guaranteed. Send $79 check to author:
Gary D. Campbell
205 Sunbird Cliffs Lane
Colorado Springs, CO 80907

MICRO COMPUTING SERVICES

CBTREE FOR C PROGRAMMERS
Provides enhanced file handling calls directly into C programs. Maintains balanced B-trees, supports unlimited number of keys and key lengths. Fast, Flexible, Efficient. No royalties. Source Code Incl. $179. MICRO COMPUTING SERVICES 2009 Hileman Road Falls Church, VA 22043 (703) 893-0118

Publishing

OHIO SCIENTIFIC/ISOTRON AND COMPATIBLE COMPUTERS! Users MUST read PEEK (65). Published monthly exclusively for above users. $19/yr U.S. $26/yr Canada; P.O. Box 347, Owings Mills, MD 21117; (301) 363-3268.

Hardware

SOFTWARE SENTINEL
A hardware key that prohibits unauthorized use. Its benefits: Unlimited backup copies; unbreakable; site licensing control; no floppy required with hard disk; transportable; transparent; pocketsize. Evaluation Kit available. PC compatible.
Rainbow Technologies, Inc.
17971 Skypark Circle, Suite E
Irvine, CA 92714 (714) 261-0228

DIAL & ORDER
24 HR. MODEM LINE
(408) 425-1371
Specializing in technical/reference books and computer supplies.
Wise Dog Computerbooks
1310 Fair Ave, Santa Cruz, CA 95060

Recruiting

CONTROL SYSTEMS SOFTWARE ENGINEER

A Challenging Opportunity in our Florida Corporate Offices

RS&H offers a challenging position with responsibility for development of systems level and applications level software for various Industrial Process Control applications.

Responsibilities will include involvement in associated hardware, and control strategy design, but a strong systems software background is the primary requirement.

Candidates should have at least three years' experience and a strong working knowledge of at least one computer operating system, assembly language, and "C". Familiarity with various DEC equipment operating systems, "C" and MACRO is particularly desirable. Experience with development of real-time software with some Process Control/Instrumentation background is desired. A BS degree in Computer Science or Engineering is a minimum requirement.

RS&H is a multi-disciplined A/E/P firm with a 43 year reputation for innovative technology applications and professional development. **RS&H** offers an attractive salary and benefits package including a prime Florida location providing a quality of life second to none.
Please send your resume in complete confidence, including salary history to **Thomas M. Hills (904) 739-2000.**

REYNOLDS, SMITH AND HILLS
Architects-Engineers-Planners, Incorporated
P.O. Box 4850
Jacksonville, FL 32201

RS&H encourages qualified minorities, women, veterans and handicapped individuals to apply

EOE/AAP

Dr. Dobb's Journal
is pleased to announce the
DDJ Classifieds

RATES: DISPLAY ADVERTISERS: Price per column inch $100. Ad must run in 3 consecutive issues.
LINE ADVERTISERS: Price per line $12 (35 characters per line including spaces). Minimum of 5 lines. 3 consecutive issues. Add $3 per line for boldface type. Add $25 if a background screen is desired.

DISCOUNTS: Frequency discounts for 6 consecutive ads (less 7%) and for 12 consecutive ads (less 15%).

MECHANICAL REQUIREMENTS: Camera ready art or typewritten copy only (phone orders excepted). Display: Specify desired size, include $20 if reduction is required. Line: Specify characters in boldface, caps. Allow 6 weeks for publication.

PAYMENTS: Full payment in advance. Check, money order, Visa, M/C, AmEx.

Run this ad in _____ issues
Size: _____ col x _____ inches or 1 col x _____ lines
Payment by: _____ check _____ m/o _____ Visa _____ M/C _____ AmEx
Card # _____ Exp Date _____
Signature _____
Print Name _____
Company _____
Address _____
City _____ St _____ Zip _____
Phone _____ Current Subscriber _____

Mail to or phone Tim Ortiz, DDJ Classifieds, 2464 Embarcadero Way, Palo Alto, CA 94303 (415) 424-0600

```
#define     MACNBR      130
#define     MACQSIZE    (MACNBR*7)
```

You will need to recompile the compiler to effect this change. However, if you are not having problems with overflowing macro storage space, you may ignore this change.

fread( ) and fwrite( )

The functions fread( ) and fwrite( ) do not return proper values. They each take a pair of arguments indicating how many items to transfer and the size (in bytes) of an item. They are supposed to return the number of "items" actually transferred, but instead they are returning the number of "bytes" transferred.

This edit fixes these problems and makes some changes for improved efficiency in write( ). Fortunately, these changes are easily made without recompiling the compiler. Look for these functions in the files FREAD.C and FWRITE.C. Using AR.COM, extract these files from CLIB.ARC and change them as indicated below:

```
FREAD.C
  ...
fread(buf, sz, n, fd) char *buf; int sz, n, fd; {
      return (read(fd, buf, n*sz)/sz);
      }
  ...

FWRITE.C
  ...
fwrite(buf, sz, n, fd) char *buf; int sz, n, fd; {
      if(write(fd, buf, n*sz) == -1) return (0);
      return (n);
      }
  ...
write(fd, buf, n) int fd, n; char *buf; {
      char *cnt; /* fake unsigned */
      cnt = n;
      while(cnt--) {
         Uwrite(*buf++, fd);
         if(Ustatus[fd] & ERRBIT) return (-1);
         }
      return (n);
      }
```

Then compile them and put them in CLIB.REL as indicated in the notes at the end of this article. Finally, using AR.COM, put them back into CLIB.ARC.

Uputsec( )

The function Uputsec( ) in CSYSLIB always assumes that the sector being written is at the end of the file. Consequently, it always initiates the next sector with the value 1A. This is bad if you are using random access techniques, because it overwrites existing data with 1A bytes.

This fix makes Uputsec( ) realize that it is not necessarily at end of file. Using AR.COM, extract CSYSLIB.C from CLIB.ARC and change it as indicated below:

```
CSYSLIB.C
  ...
Uputsec(fd) int fd; {
   if(fflush(fd)) return (NO);
   Uadvance(fd);
   if(Ustatus[fd]&EOFBIT||Usector(fd, RDRND))
      pad(Ubufptr[fd], CPMEOF, BUFSIZE);
   return (YES);
   }
```

Then compile it and put it in CLIB.REL as explained in the notes. Finally, using AR.COM, put it back into CLIB.ARC.

hier1( )

A problem in hier1( ) (file CC31.C) causes the operators += and -= in expressions like i += p (where p is a pointer) to erroneously assign to p instead of i. If p were a local pointer, then the assignment is made to the address corresponding to the value of i, thus corrupting the system. This problem occurred because, in the effort to assign attributes to a subexpression from included primitives and subexpressions, nothing was done to isolate the attributes of the receiving field.

This is fixed by allocating a short temporary array lval3[2] for passing to store( ). Using AR.COM, extract everything from CC.ARC (or just CC3.C, CC31.C, CC32.C, and CC33.C if you already have CC?.REL files). Change hier( ) in CC31.C as indicated below:

```
CC31.C
  ...
hier1(lval) int lval[ ]; {
   int k,lval2[8], lval3[2], oper;
   ...
   if(k==0) {
      needlval( );
      return 0;
      }
   lval3[0] = lval[0];
   lval3[1] = lval[1];
   if (lval[1]) {
      if(oper) {
         push( );
         rvalue(lval);
         }
   ...
   store(lval3);
   return 0;
   }
```

Then compile the compiler (only part 3 if you already have CC?.REL files for the other parts) and link the parts with CLIB.REL, giving a new CC.COM. Finally, replace CC31.C in CC.ARC.

Leading Blanks

Frank Hayes has reported that because CP/M Plus doesn't load the command tail into its buffer (at 80H)

with a leading blank, as CP/M 2.2 does, the Small-C function Uparse( ) misses a character, often causing it to fail to recognize a redirection specification.

This correction to Uparse( ) removes the assumption of a leading blank, making Uparse( ) work correctly with either CP/M 2.2 or CP/M Plus. Using AR.COM, extract CSYSLIB.C from CLIB.ARC. Change the first few lines in Uparse( ) to read as follows:

```
CSYSLIB.C
  ...
Uparse( ){
      char *count, *ptr;
      count = 128;
      ptr = Ualloc((count = *count&255)+1, YES);
      strncpy(ptr, 129, count);
      Uvec[0]=Uarg1;
  ...
```

Compile and assemble CSYSLIB and replace it in CLIB.REL. Using AR.COM, also replace the source in CLIB.ARC. Finally, compile, assemble, and link any Small-C programs that run under CP/M Plus.

Notes

(1) In the source statements shown above, the capital letter "U" prefixes many global variable and function names. Older copies of Small-C used the underscore character "_" in this position to avoid conflicts with user-declared names. Because some versions of Macro-80 will not handle a leading underscore on external references, underscores were changed to the letter "U" to make them acceptable to Macro-80. There is no significance to the use of upper case except that it stands out in the source listing as a uniqueness prefix instead of as part of the name proper. In making corrections, you should use "U" or "_" according to the usage in your existing runtime library.

(2) The commands to replace a module in CLIB.REL are:

```
LIB80
*NEWLIB=
*CLIB<..prevmodule>
*module
*CLIB<nextmodule..>
*/E
```

"Prevmodule" is the name of the module preceding the one being replaced, "module" is the one being replaced, and "nextmodule" is the one following the one being replaced. This will leave the old library named CLIB.REL and create a new one named NEWLIB.REL. When you are sure the new library is okay, delete the original library and rename NEWLIB.REL to CLIB.REL. To find the order of the modules in the original library or to verify the new library, execute the following commands:

```
LIB80
*libname/L
*^C
```

"Libname" is the name of the library being checked.

Addendum—Small-Mac Assembler

A companion assembler to the Small-C compiler is now available. Small-Mac is a complete relocatable macro assembler package written in Small-C, for use with the Small-C compiler. It runs on 8080/Z80 machines under CP/M. Its features include low cost, ease of use, CPU adaptability, source code distribution, and educational value.

The following programs are included:

MAC	macro assembler
LNK	linkage editor
LGO	load-and-go loader
LIB	library manager
CMIT	CPU configuration utility
DREL	dump relocatable files

MAC is a two-pass, table-driven, relocatable macro assembler. It "learns" the target machine from a machine instruction table (MIT), which is created with a text editor and compiled with the CMIT configuration utility. It generates relocatable object modules in the 8-bit Microsoft format. MAC is invoked with a simple command syntax and issues descriptive error messages.

LNK combines specified object modules with modules found in specified libraries to create executable programs. Its default output is a standard .COM file. However, it can generate "load-and-go" (.LGO) files for execution at any desired address.

LGO loads and optionally executes .LGO files. It provides a convenient way to invoke operating system extensions at cold start time.

LIB builds, maintains, and lists the contents of LNK compatible libraries.

CMIT compiles machine instruction tables, lists them, and optionally configures the assembler with the resulting object table.

DREL produces a formatted dump of .REL and .LIB files.

Small-Mac instruction operands may contain expressions of any complexity. Expression operators and precedence rules follow the C language. The Small-Mac macro facility is easy to learn, remember, and use. Conditional assembly and repeat pseudo-ops are not supported at this time.

Small-Mac comes on two 8-inch SSSD diskettes containing both source and object files. A 64-page manual is included. Send $30 (plus $4 for postage and handling, plus $6 for overseas air mail) to:

J. E. Hendrix
Box 8378
University, MS 38677

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 193.**

Asynchronous Protocols

by David W. Carroll

From its quiet beginnings in 1977 in Ward Christensen's MODEM program for data exchanges between dissimilar CP/M 8-bit computers, the XMODEM communications protocol has grown until it is now supported by almost every commercial communications package for 8- and 16-bit micros, by numerous public domain terminal programs, by most remote bulletin board and file exchange systems, and even by a recent implementation on CompuServe. But is XMODEM the only answer? This article focuses on the state of currently available asynchronous protocols and on the reasons for their development.

Why Protocols?

Software communications protocols are often necessary to allow communications between computer systems. Predefined protocols are required in many cases to allow reliable, error-free data communications between dissimilar computer systems. Lack of hardware DMA or interrupt capability, error checking and correction of data, binary file transmission, use of different microprocessors and disk operating systems, and provision for micro-to-mainframe links, multiple file transfers, and unattended operation are several specific reasons for using protocols.

David W. Carroll, P.O. Box 699, Pine Grove, CA 95665.
Copyrght © 1985 by David W. Carroll.

Background

Most microcomputer systems support asynchronous serial communications. This means that each byte of data is sent as a self contained unit; the data flow is made up of discrete characters, each completely independent from the others. To allow the receiving computer to detect and interpret incoming data, a start bit is sent, followed by a given number of data bits making up the character (usually 5, 7, or 8), followed by one or two stop bits. Bell Laboratories developed this technique in 1924 for use in teletypewriter communications systems. The receiving system must resynchronize with each character sent.

In the 1950s, military and commercial teletypewriter systems implemented a simple method of error checking by adding a bit to the data to indicate the parity of the data bits making up each character. Parity checking indicates to the receiver that the number of "1" bits in the character should be either odd or even (hence "odd" and "even" parity schemes). Parity checking provides about a 94 percent probability of detecting an error in received data. There is no method for correcting a detected parity error.

A 5-bit code (like Baudot) allows for a total of 32 characters (58 characters with the figures/letters mode-shifting scheme used in telex systems) and is still used in telex communications today. Many other character-coding schemes have been developed, but the 7-bit United

A status report on what Bell Labs, with some help from Ward Christensen and others, hath wrought.

States American Standard Code for Information Interchange (USASCII or simply ASCII), defined in the ANSI X3.1 standard of 1967, is the most widely used standard for micros. This code has 128 possible characters and includes numbers, upper- and lower-case English alphabet, punctuation, and control codes.

The 7-bit ASCII code, with or without parity checking, has long been a standard for time-shared mainframe remote terminal communications, mainly because the Teletype ® model 33 machine was the most common time-sharing terminal during the 1960s and 1970s.

Simple Protocols

The advent of microcomputers and low-cost modems in the past 10 years has generated a need for a simple, inexpensive and reliable way to exchange program and data files containing 8-bit binary data, as well as text files made up of standard ASCII characters. The IBM PC further expanded this need with its extended 8-bit character set made up of 256 characters. As more and larger data files are being transmitted over ordinary telephone lines, more elaborate error detection and automatic correction are also required.

In asynchronous time-sharing remote links, data is sent arbitrarily by the transmitting system, without regard for the capability of the receiving system to accept it. When the character is sent on to a terminal simply for display or for storage in a memory buffer, this does not present a problem. However, when files of data are transferred, the receiving system periodically must dump the data to a disk file. In many hardware configurations, this requires all of the system's resources; it can no longer handle incoming data during the dump period. Thus, to avoid data loss the receiver must have some way to halt the data flow from the sender.

Some simple protocols have been developed to allow unformatted text transmission between mainframes and microcomputers. Three basic methods control the data flow: handshaking, delay, and interruption. Use of handshaking involves sending a character (usually a carriage return) at the end of each line and requires the return of a character (usually a line feed) before flow resumes. The second alternative is for the sending system to delay a predetermined number of seconds after transmitting each line of text to allow the receiving system to empty its data buffer when required; this method works, but it can slow down communications dramatically. The interruption method has the system send data until it receives an interrupt character then pause until it receives the resume character. This protocol is perhaps the most efficient because it allows continuous transmission until the buffer is full, ensuring maximum throughput.

CR-LF Protocol

One common file transfer method used to prevent buffer overflow in time-sharing systems is to transmit one ASCII line of text at a time, including the carriage return character, then to wait for the receiver to send a line feed as acknowledgement. Often a delay of several seconds is added before the next line is transmitted. CR-LF is the most common example of the handshaking technique, although prompt sensing is another popular method used for uploading to an editor or BASIC interpreter.

XON—XOFF File Capture Protocol

Possibly the most common text file transfer protocol in use today is the file capture or XON–XOFF–protocol offered on almost all time-shared mainframe computers. This protocol detects two characters—usually ASCII DC1 and DC3 (^Q and ^S). XOFF or ^S pauses transmission and XON or ^Q resumes transmission. This protocol, another holdover from teletypewriter terminals, was used on teleprinters for paper tape operations.

Today, this file capture mode is used to transfer text files between dissimilar computer systems. Data is

Does your **ISAM** run on **IBM, APPLE, DEC** and **AT&T** computers? **c-tree** does, and you only **BUY IT ONCE!**

The company that introduced micros to B+ Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B+ Tree based file handlers. With c-tree™ you get:

- complete C source code written to K&R standards of portability
- high level, multi-key ISAM routines and low level B+ Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

$395 COMPLETE

Specify format:
5¼" PC-DOS 3½" Mac
8" CP/M® 8" RT-11

for VISA, MC or COD orders, call
1-314-445-6833

c-tree™ BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

Access Manager and CP/M are trademarks of Digital Research, Inc. Apple is a trademark of Apple Computer, Inc. c-tree and the circular disc logo are trademarks of FairCom IBM is a trademark of International Business Machines Corporation DEC is a trademark of Digital Equipment Corporation ©1984 FairCom

Circle **no. 37** on reader service card.

captured to a memory buffer, and XOFF is sent when the buffer is nearly full. When transmission stops, the data in the buffer is written to disk. Care must be taken to ensure that the text file does not contain any ^Q or ^S characters (as in WordStar files).

This protocol is supported by CompuServe, The Source, all three major data networks, and most time-sharing host computers and bulletin board systems. It supports only parity checking and 7-bit ASCII file transfers on most systems.

HEX File Conversions

Many older host computers and front-end communications processors do not support 8-bit data transfers over asynchronous links. How then can these systems (like CompuServe until recently) send binary data files?

A program converts the binary data to ASCII characters representing the hexidecimal value of the data. Thus, each 8-bit data byte ends up as two ASCII characters. For example:

Binary		ASCII
0110 1011	→	6B
1110 0001	→	E1
0011 1111	→	3F

These "characters" can be sent easily as text—but what about errors? To resolve this problem, a line format is used that includes a checksum of the characters, similar to Intel Hex format (CP/M-80, etc.). Obviously, this method is both inefficient and time consuming: it involves translations at both ends and more than doubles the transmitted size of the original file.

CompuServe supports this protocol, but it is becoming less popular than some others, like XMODEM.

XMODEM

Determined to solve the problem of file transfer, Ward Christensen, a skilled programmer in Chicago, designed a block-oriented file transfer technique that allows for text *and* binary file transfer with simple, yet effective, error checking. His first version of the program, called MODEM, was written in September 1977 and appeared on CP/M User Group Disk 25. It allowed two microcomputer users, each running the CP/M 80 operating system, to exchange both ASCII text files and 8-bit binary .COM files by modem or direct connect. (In the early days, there were many different types of CP/M systems, and most disk formats were not compatible, so directly transferring files over a communications link was often the only way to move programs from one type of computer to another.)

MODEM protocol transfers are made up of blocks of 132 bytes: four header bytes, consisting of a start of header (SOH) ASCII character, the 8-bit block number, the one's complement of the block number, and the 8-bit checksum, plus 128 data bytes. The receiver controls the sending of each block and any necessary retransmissions by sending an acknowledge (ACK), negative acknowledge (NAK), or cancel (CAN) ASCII code in response to each block. Timeouts are used as a further check of transmission accuracy and for resynchronization.

By 1979, BYE, an unattended system control program, and XMODEM, the unattended version of MODEM, were operating in bulletin boards across the country. Most file transfer systems were based on MODEM, BYE, and XMODEM, as well as on the programs written (in part by Ward Christensen) for one of the first microcomputer-based Computer Bulletin Board System (CBBS) ® in Chicago.

A large number of "improvements" to the first MODEM program have been written, and dozens of commercial and public domain programs now support the XMODEM protocol. Recently, the basic protocol definition was expanded to include optional Cyclic Redundancy Check (CRC) error checking, an improvement on the simple checksum method used in the original programs. CRC ensures a 99.99 percent probability of detecting errors. The CRC polynomial is specified by the CCITT international telecommunications standards organization.

MODEM7 Protocol

Mark M. Zeigler and James K. Mills wrote the MODEM7 CP/M 80 program in 1980. The MODEM7 protocol is an adaptation of the original MODEM/XMODEM protocol defined by Ward Christensen. The MODEM7 program adds the ability for wildcard and multiple file transfers to the basic XMODEM protocol, as well as optional CRC error checking.

Bruce R. Kendall translated the MODEM7 program to 8086 code for CP/M 86 and MSDOS in 1983; it is still used by many as a primary communications program. The MODEM7 protocol is supported by the FIDO bulletin board software.

Interestingly, Crosstalk and Crosstalk XVI (probably the most popular commercial communications program) do not properly support XMODEM transfers to all systems and do not support CRC mode at all. A public domain subprogram is available for 16-bit systems to correct this problem.

KERMIT Protocol

KERMIT (the protocol, not the frog) was developed at Columbia University and released to the public domain to allow efficient transmission of 8-bit binary data on systems that are hardware limited to 7-bit ASCII. The KERMIT method encodes the binary data into acceptable ASCII characters for transmission, but it uses more than four bits of each character (see the HEX scheme above) and uses a block rather than a line error-checking protocol to achieve much greater efficiency. KERMIT also allows multiple file transfers and master/slave operations.

KERMIT is available for most microcomputers and many mainframes at little or no cost. It presently is supported by the FIDO bulletin board system and Telois commercial micro communications software. It may become a widely used standard in the near future.

Microcom Networking Protocol

The Microcom Networking Protocol (MNP) is an interesting attempt by a commercial company to define an industry standard. Microcom, maker of modems and communications software, defined their protocol in 1982. It has been accepted by IBM, Tymnet, Telenet, and some others as a standard. Also, the data networks accept it as a way to provide an error-

free "local" link to their access nodes.

MNP is used in Microcom's ERA 2 communications software and in IBM's Personal Communications Manager program (written by Microcom). It is also supported by the latest version of RBBS-PC (12-4A1).

CompuServe A & B Protocols

CompuServe has defined two protocols for use with its VIDTEX service and provides software to support them on several computers, including most TANDY (TRS-80) computers, the Apple, and the IBM-PC.

CompuServe A protocol, originally developed to allow binary file transfers from the mainframes used by the service, was later updated to the current protocol. It is used to upload and download binary files. CompuServe B protocol is used primarily with Tandy (Radio Shack) TRS-80 computers for file transfers and support of full-color video graphics (VIDTEX) capabilities.

CompuServe also supports BIN and HEX file 7-bit ASCII transfers in modified Intel Hex format.

ANSI X3.28 Protocol

With the demand for reliable communications, it is surprising that the ANSI X3.28 protocol (originally defined in 1971) was not implemented on microcomputers until this past year. The X3.28 protocol is similar to MODEM7 and KERMIT. It provides block transmission with CRC-type error checking, remote system master/slave control, and multiple file transfers.

Artisoft Inc. of Tucson, Arizona, has included X3.28 protocol support in its Envoy-PC communications software, as well as file capture and XMODEM. Recently, X3.28 has received attention as a possible protocol for use in local area networks.

Specialized Protocol Software

Several software vendors have developed their own protocols for use between systems running their programs. Some of these include the following.

Crosstalk

Crosstalk and Crosstalk XVI by Microstuf both support a 256-byte block, error-checking protocol (also called CLINK) similar to MODEM7 that allows multiple file transfers. Later versions of Crosstalk also support XMODEM.

Smartcom

Smartcom and Smartcom II support the Hayes Verification Protocol, a special error-checking protocol designed by Hayes Microcomputer products for file transmission. Smartcom II also supports the XMODEM protocol.

Telink

The Telink protocol is supported by the Telink program and Minitel public domain communications program, written by Tom Jennings, and by the FIDO bulletin board system, also authored by Jennings. All programs support MODEM7 and XMODEM as well.

Move-It

Move-It from Woolf Software uses a proprietary "packet" protocol with 16-bit checksums, message numbering, and remote multiple file transfer capability. Move-It does not support XMODEM.

MITE

MITE (Mycroft Intelligent Terminal Emulator) from Mycroft Labs supports all of the above protocols (except Telink) plus the following:

- XMODEM checksum and CRC
- MODEM7 checksum and CRC (called XMODEM/Batch)
- MITE batch block protocol (see *DDJ*, August 1982)
- IBM PC (text protocol from IBM Asynchronous Support Program)
- TEXT (simple Hex transfer protocol)

ASCOM

The ASCOM program from Dynamic Microprocessor Associates supports XMODEM (called CPMUG) as well as two checksum block protocols, BLOCK and BLOCKV.

Summary

A number of asynchronous protocols have evolved to fill the need for reliable file transfers of both text and binary data between various types of mainframes and microcomputers. The most widely used standard is the XMODEM protocol originally defined by Ward Christensen, but even this protocol has four possible formats (checksum/CRC and single file/batch). Most commercial and public domain file transfer programs support at least the basic checksum, single file version of the XMODEM protocol. XMODEM is simple, fairly unambiguous, and easily implemented in a variety of languages, from C to BASIC.

Transfers from 7-bit mainframe host systems are still a clumsy and time-consuming process. The KERMIT protocol, which improves the efficiency of these transfers, is now available in a number of formats for many different computers.

Several other proprietary protocols have been developed, but none has received the widespread acceptance of XMODEM. With XMODEM so well entrenched, it is unlikely to be replaced by another protocol for several years, if at all.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 194.**

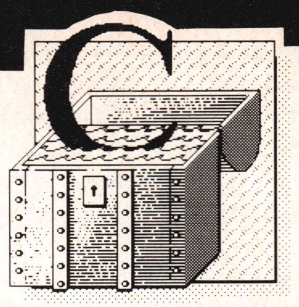
C CODE FOR THE PC
source code, of course

Concurrent C	$45
LEX	$25
YACC & PREP	$25
Small-C compiler for 8086/88	$20
tiny-c interpreter & shell	$20
Xlisp 1.4 & tiny-Prolog	$20
C Tools	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Circle **no. 7** on reader service card.

C CHEST



by Allen Holub

The main topic of this month's column is the Unix[1] "make" utility. We'll talk about how make works, review a few versions that run under MSDOS, and present the complete source for a rather stupid, but nonetheless functional, version of make for MSDOS.

However, before leaping into make, I'd first like to thank everyone who sent in sort routines. Several of the routines are quite spiffy; we'll definitely have another sorting column in the next few months.

Next, some good news for Lattice C users: they've finally cleaned up their manual. The old manual was so poorly organized as to make the compiler almost useless to someone who wasn't thoroughly familiar with Unix. Critical information was scattered over several appendixes and a myriad of technical bulletins. This wouldn't have been a problem if Lattice had published a new index with each addendum to the manual, but no such luck. As far as I can tell, the new manual uses the same text as the old one; however, it's now organized into a coherent whole with a real index (they even highlight function names in blue). If you own version 2.1x or later of the compiler, you can get a free update to the new version (2.15) and a copy of the new manual by sending Lattice your original release disks. Older versions of the compiler (including Microsoft and Lifeboat versions) can be updated for a fee ranging from $50 to $100, depending on the version. If you have an older version of the compiler and just want the new manual, send Lattice the table of contents and index from the old manual and they'll send you the new one (but no software update).

Finally, while rooting through a friend's back issues of the *PC Tech Journal*, I discovered MSDOS's undocumented SWITCHAR feature in an article by J. Eric Roskos (August 1984, page 73). It's a constant annoyance to me that command.com insists on using \ to separate directory names in a path and / as a command line switch designator (Unix programmers are used to / and -, respectively). This seemed all the more onerous when I discovered that MSDOS itself accepts either a \ or / in a path name. Only command.com and a few utility routines insist on specializing the two.

Well, it turns out that if you include the line

```
SWITCHAR = -
```

in your config.sys file, command.com will use a - instead of a / as its command line switch designator (you can replace the - with any printing character). Because / is no longer special, command.com passes it through to DOS where it's accepted as a legal path name designator. Moreover, the path printed by the "prompt $p" command will now use slashes instead of back-slashes when it displays the path!

I've no idea how this mechanism actually works, though I suspect it operates in ways similar to an environment variable. If anyone can enlighten me, please write. On page 113 of the same issue of the *PC Tech Journal*, Daniel Frank shows how to use DOS function 37 (also undocumented) to get the current switch character. Frank also presents a few SWITCHAR caveats. To summarize: RESTORE can get very confused if the SWITCHAR is set when you run BACKUP, so don't use the SWITCHAR if you're doing a backup. The same article gives a patch to RESTORE to fix this problem; the interested reader can find it there.

Unfortunately, in a fit of perversity, Microsoft elected not to support SWITCHAR in DOS version 3. The operating system spits out the message: "Unrecognized command in CONFIG.SYS." Argh. All is not lost, though. Unipress, the manufacturer of Ps-make, reviewed below, sells a program that changes the switch character, and this program *does* work with version 3.

Make

Make is one of those deceptive utilities: you can't figure out why anyone would bother to use it until you actually do use it, then you can't figure out how you got along without it. Make takes as input a makefile, which describes all the modules in a large program and the relationships between the modules. Using the makefile, make determines which modules must be recompiled at any given moment and then compiles them (and only them). It's like an intelligent batch utility that knows how C programs and C compilers work and does only those actions that are absolutely necessary.

The best way to explain make is with an example. You've an executable program called farm.exe. The original source is split up into three modules: cow.c, pig.c, and farm.c. All three modules have an #include <stdio.h> statement in them. In addition, cow.c and pig.c have an #include <animals.h> statement. Now, if you change something in cow.c, you need to recompile cow.c and then relink the new cow.obj into farm.exe. If you change something in animals.h, you need to recompile both cow.c and pig.c then relink. If you change stdio.h, you need to recompile everything. Make describes the relationships between these various files as "dependencies." The

Technical Product Information

FREE
For The Asking

See something you'd like to learn more about? Need more details?

DR. DOBB'S JOURNAL **Reader Service Card**

Name _____ **Phone** _____

Address _____

City/State/Zip _____

Expiration Date: Nov. 30, 1985 **August 1985, #107**

Please circle one for each category:

I. My firm or department is a:
- A. Systems Integrator/House
- B. Software Dev. Firm
- C. Hardware OEM or Manuf.
- D. DP, MIS or Data Service
- E. Consulting Firm
- F. Eng. or Science Lab.
- G. Other

II. My job function is:
- H. Company Mgmt./Admin.
- J. Computer Systems Mgt.
- K. Programmer/Technical Staff
- L. Consultant
- M. Engineering Mgmt. or Staff
- N. Scientific Mgmt. or Staff
- O. Other

III. Number of employees in my firm:
1. Less than 10
2. 10—99
3. 100—499
4. 500—9,999
5. 10,000—or More

IV. This inquiry is for:
- P. Immediate Purchase
- Q. Future Project

V. Purchasing Authority
(check all that apply)
- R. Recommend or Specify
- S. Final Decision-Maker
- T. No Influence

VI. I advise others about computers, on the average:
6. More Than Once-A-Day
7. Once-A-Day
8. Once-A-Week
9. Not At All

VII. I design/write software professionally
- U. Yes
- V. No

VIII. I buy computer products through:
(check all that apply)
- W. Retail Stores
- X. Mail Order Houses
- Y. On-site Direct Salespeople
- Z. All of the Above

IX. I am currently a subscriber:
- A. Yes
- B. No

Check each advertisement for corresponding number and circle below:

001	011	021	031	041	051	06
002	012	022	032	042	052	06
003	013	023	033	043	053	06
004	014	024	034	044	054	06
005	015	025	035	045	055	06
006	016	026	036	046	056	06
007	017	027	037	047	057	06
008	018	028	038	048	058	06
009	019	029	039	049	059	06
010	020	030	040	050	060	07
071	081	091	101	111	121	13
072	082	092	102	112	122	13
073	083	093	103	113	123	13
074	084	094	104	114	124	13
075	085	095	105	115	125	13
076	086	096	106	116	126	13
077	087	097	107	117	127	13
078	088	098	108	118	128	13
079	089	099	109	119	129	13
080	090	100	110	120	130	14
141	151	161	171			
142	152	162	172	181	191	20
143	153	163	173	182	192	20
144	154	164	174	183	193	20
145	155	165	175	184	194	20
146	156	166	176	185	195	20
147	157	167	177	186	196	20
148	158	168	178	187	197	20
149	159	169	179	188	198	20
150	160	170	180	189	199	20
				190	200	21

Articles

☐ **Please send me a one year subscription to D Dobb's Journal at $25.00 and bill me later.**

Note:

For quicker, more effective processing of your inquiry, please provide responses to ALL requested information.

Information that's
• Current
• In-depth
• Directed to you on specific products & services

FREE

DR. DOBB'S JOURNAL **Reader Service Card**

Name _____ **Phone** _____

Address _____

City/State/Zip _____

Expiration Date: Nov. 30, 1985 **August 1985 #107**

Please circle one for each category:

I. My firm or department is a:
- A. Systems Integrator/House
- B. Software Dev. Firm
- C. Hardware OEM or Manuf.
- D. DP, MIS or Data Service
- E. Consulting Firm
- F. Eng. or Science Lab.
- G. Other

II. My job function is:
- H. Company Mgmt./Admin.
- J. Computer Systems Mgt.
- K. Programmer/Technical Staff
- L. Consultant
- M. Engineering Mgmt. or Staff
- N. Scientific Mgmt. or Staff
- O. Other

III. Number of employees in my firm:
1. Less than 10

V. Purchasing Authority
(check all that apply)
- R. Recommend or Specify
- S. Final Decision-Maker
- T. No Influence

VI. I advise others about computers, on the average:
6. More Than Once-A-Day
7. Once-A-Day
8. Once-A-Week
9. Not At All

VII. I design/write software professionally
- U. Yes
- V. No

VIII. I buy computer products through:
(check all that apply)
- W. Retail Stores
- X. Mail Order Houses

Check each advertisement for corresponding number and circle below:

001	011	021	031	041	051	06
002	012	022	032	042	052	06
003	013	023	033	043	053	06
004	014	024	034	044	054	06
005	015	025	035	045	055	06
006	016	026	036	046	056	06
007	017	027	037	047	057	06
008	018	028	038	048	058	06
009	019	029	039	049	059	06
010	020	030	040	050	060	07
071	081	091	101	111	121	13
072	082	092	102	112	122	13
073	083	093	103	113	123	13
074	084	094	104	114	124	13
075	085	095	105	115	125	13
076	086	096	106	116	126	13
077	087	097	107	117	127	13
078	088	098	108	118	128	13
079	089	099	109	119	129	13
080	090	100	110	120	130	14
141	151	161	171			
142	152	162	172	181	191	20
143	153	163	173	182	192	20
144	154	164	174	183	193	20
145	155	165	175	184	194	20
146	156	166	176	185	195	20
147	157	167	177	186	196	20

Articles

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS
Dr. Dobb's Journal

P.O. BOX 13851
PHILADELPHIA, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS
Dr. Dobb's Journal

P.O. BOX 13851
PHILADELPHIA, PA 19101

makefile is a list of these dependencies. A makefile for the program just described is shown in Figure 1 (below).

The # designates a comment line. Farm.exe "depends on" cow.obj, pig .obj, and farm.obj. This means that if any of the files cow.obj, pig.obj, or farm.obj have been changed more recently than farm.exe, then farm.exe must be remade—by executing the line: link \lc\s\c cow pig farm,farm, con,\lc\s\lc. Similarly, if any of the files pig.c, stdio.h, or animals.h have been changed more recently than pig.obj, then pig.obj must be remade (using the command lc -ms -i\lc\ -i\lc\s\ pig). The actions associated with each set of dependencies can extend to several lines. To activate the process, you just type the command make. Make reads in the makefile and, using the information found there, figures out what to do and does it.

Make supports several options that make your life a little easier. You may use macros to save some typing. A macro is created by placing the following definition at the head of the makefile:

name = stuff

```
#
#  Make farm using the Lattice C compiler.
#

farm.exe:       cow.obj pig.obj farm.obj
                link \lc\s\c cow pig farm,farm,con,\lc\s\lc

cow.obj:        cow.c stdio.h animals.h
                lc -ms -i\lc\ -i\lc\s\ cow

pig.obj:        pig.c stdio.h animals.h
                lc -ms -i\lc\ -i\lc\s\ pig

farm.obj:       farm.c stdio.h
                lc -ms -i\lc\ -i\lc\s\ farm
```

Figure 1.
A Simple Makefile

```
#
#  Make farm using the Lattice C compiler.
#

INCLUDES = stdio.h animals.h
OBJECTS = cow.obj pig.obj farm.obj
COMPILE = lc -ms -i\lc\ -i\lc\s\

farm.exe:       $(OBJECTS)
                link \lc\s\c $(OBJECTS),farm,con,\lc\s\lc

cow.obj:        $(INCLUDES) cow.c
                $(COMPILE) cow

pig.obj:        $(INCLUDES) pig.c
                $(COMPILE) pig

farm.obj:       stdio.h farm.c
                $(COMPILE) farm
```

Figure 2.
A Makefile with Macros

C PROGRAMMERS' DBMS

db_VISTA

NEW MULTI-USER VERSION

PREFERRED over ISAM and file utilities, POWER like a mainframe DBMS, PRICE like a microcomputer utility, PORTABILITY like only C provides.

MS-DOS/UNIX

db_VISTA FEATURES

- Written in C for C.
- Fast B*-tree indexing method.
- Maximum data efficiency using the network database model.
- Multiple key records—any or all data fields may be keys.
- Multi-user capability.
- Transaction processing.
- Interactive database access utility.
- Ability to import and export dBASE II/III and ASCII files.

NO ROYALTIES
SOURCE CODE INCLUDED
MONEY BACK GUARANTEE

db_VISTA PRICE

Single user without source	$195
Single user with source	$495
Multi-user without source	$495
Multi-user with source	$990

MC/VISA/COD

Available for the Lattice, Microsoft, Computer Innovations, DeSmet and Aztec C compilers under MS-DOS, and most UNIX systems.

RAIMA CORPORATION
11717 Rainier Avenue South
Seattle, WA 98178, USA
(206) 772-1515 Telex 9103330300

CALL TOLL-FREE
1-800-843-3313
At the tone, touch 700-992.

Circle no. 91 on reader service card.

REMOVE

from your **C** programs with

PC–LINT

PC–LINT analyzes your C programs (one or many modules) and uncovers glitches, bugs, quirks and inconsistencies. It will catch subtle errors before they catch you.

PC–LINT resembles the Lint that runs on the UNIX O.S. but with more features and greater sensitivity to the problems of the 8086 environment.

- Full K&R C
- Supports Multiple Modules–finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or *prior* to scaling up to a larger memory model.
- All one pass with an integrated preprocessor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- **Introductory Price: $98.00 MC, VISA**
  (Includes shipping and handling) PA residents add 6% sales tax. Outside USA add $10.00.
- Runs on the IBM PC (or XT, AT or compatible) under DOS 2.0 and up, with a minimum of 128KB of memory. It will use all the memory available.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426

(215) 584-4261

*Trademarks: IBM *(IBM Corp.)*, PC–LINT *(Gimpel Software)*, UNIX *(AT&T)*

where name is the macro name and stuff is the text to be substituted. In the makefile the macro is expanded by $(name). Our original makefile, rewritten using macros, is shown in Figure 2 (page 97).

You can simplify the makefile even further. Notice that all .c files are converted to .obj files with the same set of actions. Make provides a mech-

```
#
#  Make farm using the Lattice C compiler.
#

INCLUDES = stdio.h animals.h
OBJECTS = cow.obj pig.obj farm.obj

.c.obj:
                lc -ms -i\lc\ -i\lc\s\ $*

farm.exe:       $(OBJECTS)
                link \lc\s\c $(OBJECTS),farm,con,\lc\s\lc $*

cow.obj:        $(INCLUDES)
pig.obj:        $(INCLUDES)
farm.obj:       stdio.h
```

Figure 3.
A Generic Makefile

```
#
#  Make farm using the Lattice C compiler.
#

farm.exe:       cow.obj pig.obj farm.obj
                link \lc\s\c cow pig farm,farm,con,\lc\s\lc

cow.obj:        cow.c stdio.h animals.h
                lc -ms -i\lc\ -i\lc\s\ cow

pig.obj:        pig.c stdio.h animals.h
                lc -ms -i\lc\ -i\lc\s\ pig

farm.obj:       farm.c stdio.h
                lc -ms -i\lc\ -i\lc\s\ farm

#
#  Null targets not needed in a real makefile:
#

animals.h:

stdio.h:

pig.c:

cow.c:

farm.c:
```

Figure 4.
An Mkfile

anism for doing this sort of repetitive action, called a "generic dependency". Consider the makefile shown in Figure 3 (page 98).

The lines:

```
.c.obj:
        lc -ms -i\lc\ -i\lc\s\ $*
```

say "to turn a .c file into an .obj file, execute the line lc -ms -i\lc\ -i\lc\s\ $*" where $* is a predefined macro that evaluates to the root portion of the file being made (the one to the left of the : on the dependency line). For example, given the dependency line:

```
foo.obj: foo.c rat.h
```

$* will evaluate to the string foo. In the case of this generic dependency, make assumes that all .obj files depend on a .c file having the same root name. So, when foo.obj is being made, a dependency on foo.c is assumed, and it need not be listed on the dependency line.

There are several commercially available make utilities that run on MSDOS. (See the review of MSDOS C compilers on page 30 for compilers that include a make utility in the standard distribution package. You can't have a make for CP/M because you can't time and date stamp a file.) I looked at three of these: Polymake (by Polytron), Ps-make (by Unipress Software), and Pmaker (by Phoenix).[2] Of these, Polymake (at $99) is both the least expensive and the most complete implementation of make. Ps-make (at $179) is missing some of the features of Polymake. Pmaker, the Phoenix product (at $195), is both the most expensive and the most limited of the three versions. The Pmaker manual is extremely sparse. Pmaker supports virtually no command line options and only a stripped-down set of internal utilities (no predefined macros, no generic dependencies). It does come with a utility that creates makefiles (by going into your source code and looking for #include statements), but because my makefiles are often a little weird, I found this utility pretty useless. Frankly, I don't think that Pmaker is worth the money. In fact, the version of make presented at the end of this column is almost as powerful as the Phoenix product—and it's free. I'll therefore limit my remarks to Polymake and Ps-make.

Polymake

Polymake supports all the make features described above. It has 13 command line options:

-a
Rebuild all targets, even if time and date say that you don't have to.

-b <file>
Look for a file called builtins.mak—Polymake supports a default actions file in which you define things like .c.obj. or macros that are used a lot.

-c
Create a batch file rather than doing the action—this lets you use Polymake with DOS version 1.

-d
Echo an execution trace as actions are performed—debug mode.

-f <file>
Use <file> as the makefile instead

Now for the IBM PC, XENIX, UNIX, VMS...

WINDOWS FOR C™

Advanced Screen Management
Made Easy
A video tool kit for all screen tasks

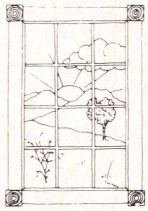
- Pop-up menus and help files
- Unlimited files and windows
- Rapid screen changes
- Logical video attributes
- Complete color control
- Horizontal and vertical scrolling
- Word wrap
- Highlighting
- Plus a library of over 65 building block subroutines

So easy to learn and easy to use, you'll wonder how you ever managed without it.

Provides application portability between the IBM PC and XENIX, UNIX, VMS or any terminal-based system.

Full support for IBM PC/XT/AT and compatibles; Lattice C, C1-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS); PC/XENIX. Source version available for Unix and other OS.

NEW Ver. 4.0
Logical attributes
Easier menus
New pop-up functions
WINDOWS FOR C
PCDOS
(specify compiler) **$245**
PC/XENIX $495
UNIX and other OS Call
Full Source Available



Vermont Creative Software
21 Elm Ave.
Richford, VT 05476
802-848-7738, ext. 31,

Master Card & Visa Accepted
Shipping $2.50
VT residents add 4% tax

Trademarks - UNIX, AT&T; XENIX, Microsoft; VMS, DEC.

Circle **no. 27** on reader service card.

of "makefile."

-i

Ignore errors returned from the programs.

-k

Keep working and ignore error codes returned from a process. (Like the Unix version, if any program invoked by Polymake returns a value other than 0, Polymake terminates. This can be annoying if you want to compile 85 modules and go to lunch, redirecting all the error messages into an error log file. The -i option suppresses this automatic termination. From reading the manual, I'm not clear about the differences between -i and -k. I've been using -k.)

-n

Print out the various actions but don't actually execute them—this is useful for debugging a makefile.

-p

Print out a representation of the makefile after it's been interpreted.

-q

Don't print out error messages from commands.

-r

Don't use any builtins.mak files (see option -b above).

-s

Don't echo commands (silent).

-t

Touch. (Perhaps the most useful option supported by make, -t is for the situation where you change a comment in stdio.h and now make thinks that all 97 modules of your program monster.exe have to be recompiled; make -t modifies all the times and dates for files to be made so that they look as if they've been recompiled, but it doesn't actually recompile them. Polymake also comes with a touch utility (invoked from MSDOS with "touch <file list>") that sets the time and date of the indicated files to the current time and date.)

There's also a rich set of predefined macros:

$(name)	Expand macro "name"
$$	$ sign
$@	Name of current target
$*	Root of current target
$<	Source being used to make current target
$?	All sources that have been modified more recently than current target
$( )	Null string
$(mflags)	Command line flags (so that make can invoke itself recursively)
$(cwd)	Current working directory

Finally, Polymake supports several internal options, as well as generic dependencies. Although I won't describe all of them here, one is quite useful: the local input script. This feature is for programs, such as librarians and linkers, that sometimes need to use input files rather than the command line. For example, you can have so many files to link together that they won't all fit on a DOS command line. In this case, you put the command line into a file then call link with @file given on the command line. Local input scripts let Polymake create these input files for you. The example given in the manual illustrates the process pretty well:

```
text.exe : t1.obj t2.obj t3.obj
```


Tools for the Programmer from Blaise Computing

Save Up To $130 On These Special Offers!

TOOLS & TOOLS 2

For C or Pascal

For a limited time, pick up both packages and save $50 off our regular list price. The C version comes with libraries for the Lattice, Computer Innovations and Microsoft (version 2.03 and 3.00) compilers. The Pascal version supports IBM and Microsoft Pascal. ***$175.***

VIEW MANAGER With Source

All libraries are included. Please specify C or Pascal. Regular $425. Save $130. ***$295***

Blaise Computing provides a broad range of fine programming tools for Pascal and C programmers, with libraries designed and engineered for the serious software developer. You get clearly written code that's fully commented so that it can serve both as a model and also be easily modified to grow with your changing needs. Our packages are shipped to you complete with comprehensive manuals, sample programs and source code. None of the programs are copy-protected.

FOR C AND PASCAL PROGRAMMERS:

TOOLS ◇ $125

Extensive string and screen handling, graphics interface and easy creation of program interfaces. Includes all source code.

TOOLS 2 ◇ $100

Memory management, general program control and DOS file support. Interrupt service routine support. Includes all source code.

VIEW MANAGER ◇ $275

General screen management. Create data entry screens that can be easily manipulated from your application program. Block mode data entry and retrieval with fast screen access.

VIEW LIBRARY Source ◇ $150

Source code to the VIEW MANAGER library functions.

ASYNCH MANAGER ◇ $175

Powerful asynchronous communications library providing interrupt driven support for the COM ports. All source code included.

FOR THE TURBO PASCAL PROGRAMMER:

Turbo POWER TOOLS ◇ $99.95

Extensive string support, extended screen and window management, interrupt service routines, program control and memory management, interrupt filters. All source code included.

Turbo ASYNCH ◇ $99.95

Interrupt driven asynchronous communication support callable from Turbo Pascal. ASYNCH is written in assembler and Turbo Pascal with all source code included.

PACKAGES FOR ALL PROGRAMMERS:

EXEC ◇ $95

Program chaining executive. Chain one program from another even if the programs are in different languages. Common data area can be specified. Source code included if you're a registered C TOOLS and C TOOLS 2 user.

SPARKY ◇ $75

Run-time resident (or stand-alone) scientific, fully programmable, reverse polish notation calculator. No limit on stack size, variables or tape. Includes all standard scientific functions and different base arithmetic.

TO ORDER, call Blaise Computing Inc. at (415) 540-5441

◆ 2034 Blake Street ◆ Berkeley, CA 94704 ◆ (415) 540-5441 ◆

Circle **no. 8** on reader service card.

```
  link <@<
t1+
t2+
t3
test.exe
test.map
\lc\s\lc.lib
<
```

All text between the second and third < sign is put into a file called lis_qqq.tmp. Link is invoked with the line link @lis_qqq.tmp. The @ in this example can be replaced by any printing character. For example,

```
text.exe: t1.obj t2.obj t3.obj
  link <<<
text
<
```

will execute link <lis_qqq.tmp; lis_qqq.tmp will contain the single line "text." The temporary file is deleted when Polymake is done with it.

As you can probably tell, I like Polymake. I've been using it for a month now and haven't found any problems yet. My only complaint is that the manual doesn't have an index (which will make it hard to find anything if you're not familiar with the Unix make). On the other hand, Polymake sticks pretty close to the Unix model, so you may not need to consult the manual very often.

Ps-make

As I said earlier, Ps-make is more limited that Polymake, but it does do everything I consider to be essential: generic dependencies, $@, $*, $<, and macros. Several command line options are supported (there is some overlap with Polymake):

-f<file>
Use <file> as the makefile.
-o<file>
Use <file> as name of batch file (see -b).
-b
Create a batch file instead of executing commands.
-B
Beep before exiting.
-c
Don't convert to lower case when evaluating rules.
-d
Print file modifications times used in comparisons.
-e
Ignore various environment variables—the name of the default rules file (builtins.mak in Polymake) is kept in an environment variable.
-p
Print the generated dependency tree.
-s
Don't execute commands as they're executed.
-t
Touch.
-v
Force Ps-make to verify that a generated file exists before using it.

A MAKE Utility in C

One of the first utilities I wrote for MSDOS was the version of make given here. The program's called mk (i.e., a make with half of it missing). Mk is pretty stupid; it doesn't support macros or command line switches. On the other hand, it does do the critical job: recompile only those parts of a large modular program that need it. I was glad to have it before I got my copy of Polymake, and it is, after all, free. Mk was designed to help maintain large C programs, and it works quite well in this application, but it won't do everything that the real make can do.

Mk uses a makefile called "mkfile," which, like the real makefile, lists all the dependencies needed to make a program. If mk is invoked with a name as an argument (i.e., mk foo.exe), it will make the indicated file; that is, it looks in the mkfile for the indicated filename to the left of a colon on a dependency line and starts the make from that point. If no file is specified on the command line, the first file listed in the mkfile is made.

The mkfile itself has a more restricted syntax than a real makefile:

(1) A # sign in column 1 designates a comment line. The # must be in the leftmost column.

A FULL C COMPILER FOR $49.95

The Eco-C88 C compiler is setting a new standard for price and performance. Compare Eco-C88's performance to compilers costing up to 10 times as much:

	Seive	Fib	Defref	Matrix
Execute	12.1 sec.	43.1 sec.	13.7 sec.	21.3 sec.
Code Size	7782	7754	7772	9120
Compile-link	76 Sec.	77 Sec.	77 Sec.	92 Sec.

Eco-C88 Rel. 2.20, on IBM PC with 2 floppy disks, 256K. Benchmarks from Feb., 1985 **Computer Language.**

Eco-C88 includes:
* All operators and data types (except bit fields)
* Error messages in English with page numbers that reference the **C Programming Guide** - a real plus if you're just getting started in C.
* Over 170 library functions, including color and transcendentals
* New Library functions for treating memory as a file
* User-selectable ASM or OBJ output (no assembler required)
* 8087 support with 8087 sensed at *runtime*
* cc and "mini-make" for easy compiles (with source)
* Fast, efficient code for all IBM-PC, XT, AT and compatibles using MSDOS 2.1 or later.
* Complete user's manual

If ordered with the compiler, the C library source code (excluding transcendentals) is $10.00 and the ISAM file handler (as published in the **C Programmer's Library,** Que Corp) in OBJ format is an additional $15.00. Please add $4.00 for shipping and handling. To order, call or write:

MasterCard VISA AMERICAN EXPRESS

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220
(317) 255-6476 • 8:30-4:30

ECOSOFT INC.

Eco-C (Ecosoft), MSDOS (Microsoft), UNIX (Bell Labs), CP/M (Digital Research), Z80 (Zilog), 8086, 8087, 8088 (Intel).

Circle **no. 35** on reader service card.

C POWER
C LIBRARIES
C WINDOWS

Best You Can Get!
325 Fully Tested Functions

Best Documentation
(over 400 pages)

SIX C LIBRARIES
For IBM PC, XT, AT

All Source Code. No royalties.
51 screen handling/graphics
50 cursor/keyboard/data entry
85 superior string handling
25 system status & control
72 utility/DOS/BIOS/time/date
42 printer control

RICHLY COMMENTED
EASY TO LEARN
EASY TO MODIFY

NO MATTER WHAT ELSE
YOU HAVE
GET THESE!!

ALL 6 LIBRARIES $99.95
(ON FOUR DISKETTES)

POWER WINDOWS

PROFESSIONAL WINDOW MANAGEMENT

OVERLAYS, BORDERS, POPUP-MENUS, COLOR HIGHLIGHTING, HELP WINDOWS, STATUS-LINE, MONOCHROME OR COLOR, FILE, CURSOR, KEYBOARD CONTROL AND MORE !!

C WINDOWS: COMPLETE SOURCE CODE $99.95

ALL LIBRARIES PLUS WINDOW $159.95

Entelekon
SOFTWARE SYSTEMS
ENTELEKON 12118 KIMBERLEY
HOUSTON, TX. 77024 (713)-468-4412

VISA • MASTERCARD • CHECK

Circle no. 52 on reader service card.

(2) The dependency line comes first. It uses the format "target: dependencies". The colon is required (an error message is printed if it's not there). The dependencies are delimited from one another with white space (any combination of tabs and blanks). The dependencies must all be listed on a single line; however, any line terminated with a back-slash (\) will be continued to the next line. For example:

```
This is\
one line\
of text.
```

(3) All files listed as dependencies must be used as targets somewhere in the mkfile (i.e., to the left of a colon). A null target (one consisting of a single filename followed by a colon but no dependencies, this in turn followed by a blank line) is permitted.
(4) All lines following the dependency line, up to a blank line, are the actions executed to make the target. The blank line is required to terminate the block of actions. Any number of blank lines are permitted as a terminator, but any nonblank line following a blank line is assumed to be a new dependency line.
(5) Targets having no dependencies will always be made. For example,

```
listing:
    print foo.c bar.c rat.c
```

will always execute the print command.
(6) For the sake of comparison, all nonexistent files are considered to exist and to be very old, so a nonexistent file listed to the left of a colon will always be made. The dates and times associated with a target are updated as soon as the target is made.

Figure 4 (page 98) is a mkfile version of the makefile in Figure 1. Here the blank lines are required, as are the various null targets at the end of the file (for animals.h, stdio.h, etc.).

The listing for mk starts on page 104; mydos.h (#included on line 3) was presented last month. There are several macros at the head of the listing. #including the #define on line 6 will cause mk to generate a trace as it parses the mkfile, but it won't actually try to do anything. MAXLINE (on line 8) limits the number of characters in a line (lines continued with a terminating \ are considered to be one line). MAXBLOCK and MAXDEP (lines 9 and 10) are the maximum number of actions that can follow a dependency line and the maximum number of dependencies, respectively. MAKEFILE (line 12) is the name used for the makefile. The three macros on lines 24–26 have horrible side effects, so be careful with them (don't ever say skipwhite(++s)).

I suspect that the real make sets up a dependency tree, a multi-way tree that has the major file to be made as its root and dependencies as children. Mk doesn't work that way. Mk reads in the entire mkfile before it tries to do anything. If it gets lost (or finds a syntax error), an error message is printed and mk terminates.

The makefile is organized as a binary tree, ordered alphabetically by target filename; the tree nodes are typedefed on lines 38–47. Lnode and rnode are pointers to children in the tree. The target filename (the one to the left of the equals sign) is used as the search key and stored in the being_made field.

The dependencies and the actions following the dependencies are put into two argv-like arrays of pointers to strings. The fields depends_on and do_this are used like argv to access these arrays. Both arrays are terminated with a null entry, so the equivalent of argc isn't required.

Finally, the time field holds the file create time and date returned by DOS. The time and date are concatenated (with the date in the most significant word). This enables the time field to be used as a single number in comparisons.

Most of the routines are commented well enough to be self-documenting, but there are some exceptions. Gtime( ) (lines 97–137) gets the time and date for a file from DOS, concatenates the time and date together, and returns the concatenated result. The functions gregs( ) and dos( ), used to access DOS, were presented in

this column last month.

The assignment on lines 129–30 merits some attention. Regs.x.dx and regs.x.bx are both shorts. A short, shifted left 16 bits, evaluates to 0, so dx has to be cast into a long before the shift. Similarly, if regs.x.cx isn't cast into a long, the | will behave unpredictably, depending on whether sign extension is active. However, now we have to AND the result of the type conversion with 0xffff to defeat the sign extension (or else we may end up setting the entire top 16 bits to 1).

The function getline( ) (lines 200–251) is useful enough to merit some notice, too. It works something like gets( ) except that it recognizes line continuation (lines ending in \) and it allocates buffer space using malloc( ). I've used it in several other programs.

The final routine to look at here is make( ) (lines 302–381); all the real work performed by mk is done here. Make( ) is recursive. To make a single object, you have to make the dependent files. If any of the dependent files were remade, you then have to perform the action on the current file. When make( ) returns, the dates on the remade files will have been modified to the current date. Consequently, the time and date comparisons for the target and the dependent file (on line 339) must follow the recursive make( ) call (on line 327). If any of the dependencies were younger than the target, doaction is incremented (line 350). We don't actually do the action on line 350 because we don't want to do it several times.

Finally, mk uses a system( ) call (line 368) to execute each action line. In Unix, system( ) creates a parallel process running the shell then passes its argument through to the spawned shell. Because Unix supports true multitasking, the only real penalty involved in a Unix system( ) call is the time it takes to activate the shell.

MSDOS is another matter. System( ) actually reads a *second copy* of command.com into memory then executes the command line with this second copy. Because command.com uses about 21K and because the program being invoked takes up some space, as does mk itself (about 15K in my version), memory can disappear pretty fast. I've a bunch of memory, so this limitation hasn't been a problem, but it may be for you.

There are two solutions. The better of these works only if you aren't going to use any functions internal to command.com (more importantly, you can't run a batch file from inside mk). The Lattice compiler supplies several system( )-like procedures. In particular, the various fork( ) functions (forkv, forkl, forkvp, forklp) will execute another program directly, without the additional invocation of command.com. These functions are something of a misnomer as they don't behave like the Unix fork( ) function; however, at the cost of a little overhead (you have to extract the program name from the action), you can use a fork( ) function instead of a system( ) call to do the action. If your compiler doesn't have a fork( ) or a system( ) call, you can create one using the DOS version 2 EXEC function (0x4b). I can print a system( ) function in this column at a later date if enough people need one, but most compilers seem to come with this function.

The second solution to the space problem is more involved. Instead of doing the action from inside mk, the program can create a .bat file and exit—you can then execute the batch file from outside mk. If you're running DOS version 1, you have to use this procedure because the EXEC function isn't supported. The problem here is that the time and date associated with a target are dynamically updated by mk when the target is remade (on line 374 of the listing). This new time is used by previous invocations of the subroutine make( ) to see if they need to do an action. Because gtime( ), the routine that does the updating, looks at the real file, it can't be used for updating if this file hasn't actually been modified. So, in addition to replacing the system( ) call on line 368 with a fprintf( ) call, you need to replace the gtime( ) call on line 374 with either a representation of the current time and date or with some constant that looks like a very young time and date (i.e., 0x7fffffffL).

DeSmet C

8086/8088 Development Package **$109**

FULL DEVELOPMENT PACKAGE
- Full **K&R** C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete **STDIO** Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 AND S/W FLOATING POINT

OVERLAYS

OUTSTANDING PERFORMANCE
- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER **$50**
- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT **$35**
- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check: ☐ Dev. Pkg (109)
☐ Debugger (50)
☐ DOS Link Supt. (35)

SHIP TO: _____

_____

_____ ZIP_____

CWARE CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add $5, elsewhere add $15. Checks must be on US Bank and in US Dollars. Call 9 a.m. – 1 p.m. to CHARGE by VISA/MC/AMEX.
Street Address: 505 W. Olive, #767, (94086)

Circle **no. 57** on reader service card.

One final caveat: system( ) prepends the argument /c to the command when it executes EXEC. (The process is described in section 7 of the version 3 *DOS Technical Reference* and in the version 2 manual somewhere.) Anyway, if you've used the SWITCHAR feature described earlier, DOS will be confused by the /, so set the switch character back to / before you use mk.

Erratum: I'm embarrassed to admit that my complaints last month about the erase-screen and erase-inline functions of ANSI.SYS were unfounded. I was using puts( ) to do my output and the extra newline character came from puts( ), not from ANSI.SYS.

Notes

[1]Unix is a registered trademark of Bell Labs.

[2]Polymake, Ps-make, and Pmaker are registered trademarks of Polytron, Phoenix, and Unipress Software, Inc., respectively.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 195.**

C Chest Listing (Text begins on page 96)

```
 1: #include <stdio.h>
 2:
 3: #include <mydos.h>

 4: /*-------------------------------------------------------------------------*/

 5: #ifdef  NEVER
 6: #define DEBUG 1             /* Include for debug diags in make()     */
 7: #endif

 8: #define MAXLINE  (80*10)    /* Maximum input line length             */
 9: #define MAXBLOCK 64         /* Max number of lines in an action      */
10: #define MAXDEP   32         /* Max number of dependancies            */
11: #define COMMENT  '#'        /* Delimits a comment                    */
12: #define MAKEFILE "mkfile"   /* Name of makefile                      */
13: #define OPEN     0x3d       /* Dos function call to open a file      */
14: #define CLOSE    0x3e       /* Dos function call to close a file     */
15: #define DATETIME 0x57       /* " to get or set file's date & time    */

16: #define DEFTIME  0x0        /* The default time returned by gtime when a file
17:                             * doesn't exist.
18:                             */

19: /*-------------------------------------------------------------------------
20:  *      iswhite(c)      evaluates true if c is white space.
21:  *      skipwhite(s)    skips the character pointer s past any white space
22:  *      skipnonwhite(s) skips s past any non-white characters.
23:  */

24: #define iswhite(c)      ((c) == ' ' || (c) == '\t')
25: #define skipwhite(s)    while( iswhite(*s) )        ++s;
26: #define skipnonwhite(s) while( *s && !iswhite(*s) ) ++s;

27: /*-------------------------------------------------------------------------
28:  * The entire makefile is read into memory before it's processed. It's
29:  * stored in a binary tree composed of the following structures:
30:  * depends_on and do_this are argv-like arrays of pointers to character
31:  * pointers. The arrays are null terminated so no count is required.
32:  * The time field is a 32 bit long consisting of the date and time
33:  * fields retuned from a DOS 0x57 call. The date and time are
34:  * concatanated with the date in the most significant 16 bits and the
35:  * time in the least significant. This way they can be compared as
36:  * a single number.
37:  */

38: typedef struct _tn
39: {
40:         struct _tn      *lnode;     /* pointer to left sub-tree         */
41:         struct _tn      *rnode;     /* pointer to right sub-tree        */
```

(Continued on page 106)

Periscope ... A Direct Hit!

"A great debugger ... If you've been frustrated by programs that don't behave the same for the debugger as when running alone, or that crash altogether, you should check out Periscope ..."

PC REPORT, Boston Computer Society

"It works, and works well!! In the first day of use I finished up two weeks of problems!! Really excellent!!!"

Peter Loats, Periscope User

Periscope's differences hit home! Its breakout switch gives you control, even when interrupts are disabled. Periscope's unique power helps you solve the difficult problems. Debug device-drivers, memory-resident, and non-DOS programs.

Great for straightening out confused C pointers! Using its breakpoint power, you can stop on reads and writes to ranges of memory. Stop on registers, words, and bytes, too.

It's tough. Periscope's 16K RAM board is write-protected. Runaway programs can't touch crucial debugger code! Examine the system, safely, at any time: Periscope saves the interrupt vectors and buffers it uses, then restores them when done. It uses ROM BIOS calls for functions except file access, so DOS can't clobber itself.

It gets you moving. It's commands are similar to Debug's, so you'll master it quickly. On-line help is there when you need it. You can define the windows you want; you can design easy-to-read memory displays. Periscope supports C, Assembler, BASIC, and Pascal. Comprehensive symbol support gives you a roadmap through memory tying back to your source.

It's risk-free. Periscope is backed up by a 30-day, money-back guarantee! The board is warranted for a year. Technical Support? You get the best there is ... direct from Brett Salter, Periscope's author!

It requires. An IBM PC, XT, AT, Compaq, or close compatible; PC-DOS, 64K RAM; a disk drive and an 80-column monitor. With two monitors, Periscope's screen is displayed on the monitor not in use. With one monitor, a keystroke switches screens.

It's power you can afford. At $295, you can afford Periscope's professional power. The system includes the memory board, breakout switch, debugger software, manual, and quick reference card.

Order now! Call toll-free (800) 722-7006 to place your order or to get more information. MasterCard/VISA accepted.

Get your programs up and running;

UP PERISCOPE!

Data Base Decisions/14 Bonnie Lane | Atlanta, GA 30328/(404) 256-3860

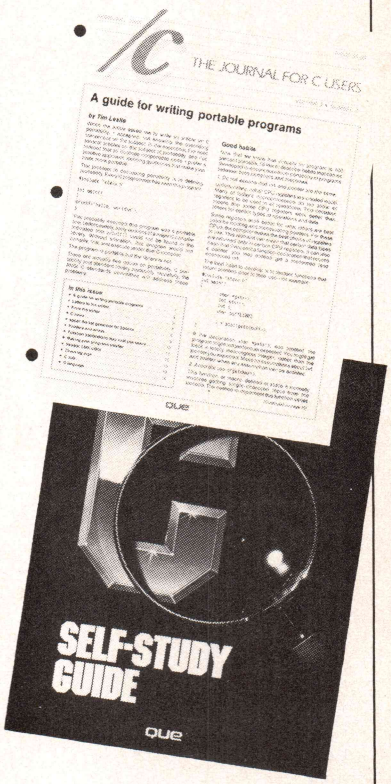
Circle **no. 31** on reader service card.

C^2 = Great Value

Free Book Bonus Subscribe Today!

Subscribe today to */c*, Que's monthly journal for C programmers, and receive FREE a copy of Que's newest C book, the *C Self-Study Guide.*

With this special offer, you'll "square away" two great C values. Not only will you gain valuable programming insight from the techniques featured monthly in */c*, but you'll also improve your basic C programming skills with the tutorial materials presented in the *C Self-Study Guide.*



Don't wait. Call today to order your */c* subscription ($60 inside U.S., $80 outside U.S.) and to receive this $16.95 value FREE. This offer expires September 30, 1985.

CALL TODAY
AND ORDER!
1-800-227-7999, ext. 66

que™

Que Publishing, Inc.
P. O. Box 50507
Indianapolis, IN 46250

Circle **no. 25** on reader service card.

```
42:         char            *being_made; /* name of file being made         */
43:         char            **depends_on;/* names of dependant files        */
44:         char            **do_this;   /* Actions to be done to make file */
45:         long            time;        /* time & date last modified       */
46: }
47: TNODE ;

48: /*-------------------------------------------------------------------------*/

49: static TNODE     *Root     = 0  ;        /* Root of file-name tree        */
50: static FILE      *Makefile      ;        /* Pointer to opened makefile    */
51: static int       Inputline = 1  ;        /* current input line number     */
52: static char      *First    = "" ;        /* Default file to make          */

53: extern char      *malloc();              /* From standard library         */
54: extern char      **getblock();           /* Declared in this module       */
55: extern char      *getline();             /* ditto                         */
56: extern TNODE     *find();                /* ditto                         */

57: /*-------------------------------------------------------------------------*/

58: char    *gmem( numbytes )
59: {
60:         /*      Get numbytes from malloc. Print an error message and
61:          *      abort if malloc fails, otherwise return a pointer to
62:          *      the memory.
63:          */

64:         extern  char    *calloc();
65:         char    *p;

66:         if( !( p = calloc(1,numbytes) ))
67:                 err("Out of memory");

68:         return p;
69: }

70: /*-------------------------------------------------------------------------*/

71: char    **stov(str, maxvect )
72: char    *str;
73: {
74:         /*      "Str" is a string of words seperated from each other by
75:          *      white space. Stov returns an argv-like array of pointers
76:          *      to character pointers, one to each word in the original
77:          *      string. The white-space in the original string is replaced
78:          *      with nulls. The array of pointers is null-terminated.
79:          *      "Maxvect" is the number of vectors in the returned
80:          *      array. The program is aborted if it can't get memory.
81:          */
82:
83:         char    **vect, **vp;

84:         vp = vect = (char **)  gmem( (maxvect + 1) * sizeof(str) );
85:         while( *str && --maxvect >= 0 )
86:         {
87:                 skipwhite(str);
88:                 *vp++ = str ;
89:                 skipnonwhite(str);
90:                 if( *str )
91:                         *str++ = 0;
92:         }

93:         *vp = 0;
94:         return( vect );
```

```
95: }

96: /*---------------------------------------------------------------------------*/

97: long    gtime( file )
98: char    *file  ;
99: {
100:        /*      Return the time and date for file.
101:         *
102:         *      The DOS time and date are concatanated to form one
103:         *      large number. Note that the high bit of this number
104:         *      will be set to 1 for all dates after 2043, which will
105:         *      cause the date comparisions done in make() to not work.
106:         *      THIS ROUTINE IS NOT PORTABLE (because it assumes a 32
107:         *      bit long).
108:         */

109:        static REGS     regs;
110:        short           handle = 0;     /* Place to remember file handle */
111:        long            time;

112:        gregs( &regs );
113:        regs.h.ah = OPEN ;              /* Open the file                 */
114:        regs.h.al = 0;                  /*      for read                 */
115:        regs.x.dx = (short) file ;

116:        if(  dos(&regs) & CARRY )
117:        {
118:                /* File doesn't exist. Return a huge date & time        */

119:                return DEFTIME;
120:        }
121:        else
122:        {
123:                handle    = regs.x.ax;
124:                regs.x.bx = handle;     /* Put file handle in BX        */
125:                regs.h.ah = DATETIME;   /* Get or set date and time     */
126:                regs.h.al = 0;          /* Get the date and time        */

127:                if( dos( &regs ) & CARRY )

128:                        err("DOS returned error from date/time request");

129:                time = ((long)( regs.x.dx ) << 16)  |
130:                                        ((long)( regs.x.cx ) & 0xffffL) ;

131:                regs.h.ah = CLOSE  ;    /* Close the file handle        */
132:                regs.x.bx = handle ;

133:                if( dos( &regs ) & CARRY )
134:                        err("DOS returned error from file close request");

135:                return time;
136:        }
137: }

138: /*---------------------------------------------------------------------------*/

139: TNODE   *makenode()
140: {
141:        /* Create a TNODE, filling it from the makefile
142:         * and return a pointer to it. Return NULL if there are no more
143:         * objects in the makefile.
144:         */

145:        char     *line, *lp;
146:        TNODE    *nodep;
147:        unsigned date, time;
```

(Continued on next page)

```
148:          /*      First, skip past any blank lines or comment lines.
149:           *      Return NULL if we reach end of file.
150:           */

151:          do{
152:                  if( (line = getline(MAXLINE,Makefile)) == NULL )
153:                          return( NULL );

154:          } while ( *line == 0 || *line == COMMENT );


155:          /*      A this point we've gotten what should be the dependancy
156:           *      line. Position lp to point at the colon.
157:           */

158:          for( lp = line; *lp && *lp != ':' ; lp++ )
159:                  ;


160:          /*      If we find the colon postion, lp to point at the first
161:           *      non-white character following the colon.
162:           */

163:          if( *lp != ':' )

164:                  err( "missing ':'" );   /* This will abort the program */
165:          else
166:                  for( *lp++ = 0; iswhite(*lp) ; lp++ )
167:                          ;

168:          /*
169:           *      Allocate and initialize the TNODE;
170:           */

171:          nodep              = (TNODE *) gmem( sizeof(TNODE) );
172:          nodep->being_made  = line ;
173:          nodep->time        = gtime( line );
174:          nodep->depends_on  = stov( lp, MAXDEP );
175:          nodep->do_this     = getblock( Makefile );

176:          return( nodep );
177: }

178: /*-------------------------------------------------------------------------*/

179: dependancies()
180: {
181:          /* Manufacture the binary tree of objects to make. First
182:           * is a pointer to the first target file listed in the
183:           * makefile (ie. the one to make if one isn't explicitly
184:           * given on the command line. Root is the tree's root pointer.
185:           */

186:          TNODE   *node;

187:          if( node = makenode() )
188:          {
189:                  First = node->being_made ;

190:                  if( !tree(node, &Root) )
191:                          err("Can't insert first node into tree !!!\n");

192:                  while( node = makenode() )
193:                          if( !tree( node, &Root ) )
```

```
194:                                    free( node );
195:                    return 1;
196:            }

197:            return 0;
198: }


199: /*---------------------------------------------------------------------------*/

200: char    *getline( maxline, fp )
201: FILE    *fp;
202: {

203:            /*      Get a line from the stream pointed to by fp.
204:             *      "Maxline" is the maximum input line size (including the
205:             *      terminating null. A \ at the end of line is
206:             *      recognized as a line continuation, (the lines
207:             *      are concatanated). Buffer space is gotten from malloc.
208:             *      If a line is longer than maxline it is truncated (ie.
209:             *      all characters from the maxlineth until a \n or EOF is
210:             *      encountered are discarded.
211:             *
212:             *      Returns: NULL on a malloc failure or end of file.
213:             *               A pointer to the malloced buffer on success.
214:             */

215:            static   char   *buf ;
216:            register char   *bp  ;
217:            register int    c, lastc;

218:            /* Two buffers are used. Here, we are getting a worst-case buffer
219:             * that will hold the longest possible line. Later on we'll copy
220:             * the string into a buffer that's the correct size.
221:             */

222:            if( !(bp = buf = malloc(maxline)) )
223:                    return NULL;

224:            while(1)
225:            {
```

(Continued on next page)

Thunder Software

• **The THUNDER C Compiler** - Operates under the APPLE Pascal 1.1 or 1.2 operating system or ProDOS. Create fast native 6502 programs to run as stand alone programs or as subroutines to Pascal programs. A major subset of the C defined by K & R. Includes a users guide, newsletters, Macro preprocessor, runs on APPLE][,][+, //e, //c. Source code for libraries is included. **Only $49.95**

• **ASSYST: The Assembler System** - A complete 6502 editor/ assembler and lister for APPLE DOS 3.3 or ProDOS. Menu driven, excellent error trapping, users guide, demo programs, source code for all programs! Great for beginners. **Only $29.95**

• **THUNDER XREF** - A cross reference utility for APPLE Pascal 1.1. XREF generates cross references for each procedure. **Only $19.95**

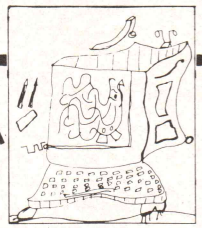
• **PDL** - Pascal Development Library. SCREENIO, DISKIO, CONVERSIONS, GRAPHICS, PLOT, MISC Units for Pascal programming. A must. **Only $29.95**

• **PROPLOT** - Easy and powerful graphing package for plotting data. **Only $24.95**

• **LINKIT** - Structured APPLESOFT processor. No more line numbers. **Only $39.95**

POB 31501 Houston Tx. 77231
713-728-5501
Visa, Mastercard, COD, checks accepted
Add $3.00 for P&H

Circle **no. 100** on reader service card.



C-BUNDLE $99

VIEW: CRT Based Disk Diagnostic
EZZAP: ROM Burning Utility
includes schematic
C-PACK: Utilities in C
C-Games: User Modifiable Maze Game

All are written in C, include Source Code, and available separately.

CP/M TM Digital Research
MSDOS TM Microsoft
ISIS TM Intel Corp

Western Wares 303-327-4898
Box C • Norwood, CO 81423

Circle **no. 130** on reader service card.

Now With Windowing!
$49.95 Basic Compiler

MTBASIC

Features:

Multitasking	**Windowing**
Handles interrupts	**Interactive**
Fast native code	**Compiles quickly**
Floating point	**No runtime fee**

MTBASIC is a true native code compiler. It runs *Bytes*'s Sept. '81 seive in 26 seconds; interpreters take over 1400 seconds! Because MTBASIC is multitasking, it can run up to 10 Basic routines at the same time, while displaying ten separate windows. Pop-up/down menus are a snap to implement.

MTBASIC combines the best of interpreters and compilers. To the programmer, MTBASIC appears to be an extremely fast interpreter. MTBASIC compiles a typical 100 line Basic program in 1 second, yet it generates blindingly fast code. No more waiting for long compiles.

AVAILABLE for CP/M (Z-80) and PC-DOS systems.

ORDERING: Specify format when ordering. We accept Visa, MC, checks and COD. Send $49.95 plus $3.50 shipping and handling ($10 overseas) to:

SOFTAID, Inc.
P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

Circle **no. 88** on reader service card.

```
226:                    /* Get the line from fp. Terminate after maxline
227:                     * characters and ignore \n following a \.
228:                     */

229:                    Inputline++;                    /* Update input line number     */

230:                    for( lastc=0; (c = fgetc(fp)) != EOF && c!='\n'; lastc = c)
231:                            if( --maxline > 0 )
232:                                    *bp++ = c;

233:                    if( !( c == '\n' && lastc == '\\' ) )
234:                            break;

235:                    else if( maxline > 0 )                  /* erase the \ */
236:                                    --bp;
237:            }
238:            *bp = 0;


239:            if( (c == EOF && bp == buf)  ||  !(bp = malloc((bp-buf)+1))  )
240:            {
241:                    /*      If EOF was the first character on the line or
242:                     *      malloc fails when we try to get a buffer, quit.
243:                     */

244:                    free(buf);
245:                    return( NULL );
246:            }

247:            strcpy ( bp, buf );     /* Copy the worst-case buffer to the one  */
248:                                    /* that is the correct size and ...       */
249:            free   ( buf );         /* free the original, worst-case buffer,  */
250:            return ( bp  );         /* returning a pointer to the copy.       */
251: }

252: /*-------------------------------------------------------------------------*/

253: char    **getblock( fp )
254: FILE    *fp;
255: {
256:            /* Get a block from standard input. A block is a sequences of
257:             * lines terminated by a blank line. The block is returned as
258:             * an array of pointers to strings. At most MAXBLOCK lines can
259:             * be in a block. Leading white space is stripped.
260:             */

261:            char    *p, *lines[MAXBLOCK], **blockv = lines ;
262:            int     blockc = 0;

263:            do {
264:                    if( !(  p = getline(MAXLINE,Makefile)  ))
265:                            break;

266:                    skipwhite(p);

267:                    if( ++blockc <= MAXBLOCK )
268:                            *blockv++ = p;
269:                    else
270:                            err("action too long (max = %d lines)", MAXBLOCK);

271:            } while( *p );


272:            /*      Copy the blockv array into a safe place. Since the array
273:             *      returned by getblock is NULL terminated, we need to
```

```
274:            *       increment blockc first.
275:            */

276:           blockv = (char **) gmem( (blockc + 1) * sizeof(blockv[0]) );
277:           movmem( lines, blockv, blockc * sizeof(blockv[0]) );
278:           blockv[blockc] = NULL ;

279:           return blockv;
280:   }

281:   /*-------------------------------------------------------------------------*/


282:   err( msg, param )
283:   char    *msg;
284:   {
285:           /*      Print the error message and exit the program.
286:            */

287:           fprintf(stderr,"Mk (%s line %d): ", MAKEFILE, Inputline );
288:           fprintf(stderr, msg, param );
289:           exit(1);
290:   }

291:   serr( msg, param )
292:   char    *msg, *param;
293:   {
294:           /*      same as err() except the parameter is a string pointer
295:            *      instead of an int.
296:            */

297:           fprintf(stderr,"Mk (%s line %d): ", MAKEFILE, Inputline );
298:           fprintf(stderr, msg, param );
299:           exit(1);
300:   }

301:   /*-------------------------------------------------------------------------*/

302:   make(what)
303:   char *what;
```

(Continued on next page)

NEW! Advanced Trace86™

Symbolic Debugger & Assembler Combo

- Full-screen trace with single stepping; Even backstepping!
- Write & Edit COM & EXE programs
- Conditional breakpoints (programmable)
- Switch between trace and output screen; Or set up two monitors
- 8087, 80186, 80286, 80287 support
- Write labels & comments on code
- Polish hex/decimal calculator
- and more . . . Priced at $175.00

To order or request more information contact:

Morgan Computing Co., Inc.
2520 Tarpley Rd. Suite 500
(214) 245-4763 Carrollton, TX 75006

Circle **no. 128** on reader service card.

FORTRAN PROGRAMMERS

Discover why you should be using **F77L** the complete implementation of the ANSI FORTRAN 77 Standard for the IBM PC and compatibles.
If you are serious about your FORTRAN programming, you should be using F77L.
$477

Lahey Computer Systems, Inc.
31244 Palos Verdes Drive West, Suite 243
Rancho Palos Verdes, California 90274
(213) 541-1200
Serving the FORTRAN community since 1969

Circle **no. 104** on reader service card.

NOW C HERE!
CROSS SOFTWARE for the NS32000

Also Available for IBM PC

INCLUDES:

★ Cross Assembler ★
★ Cross Linker ★
★ Debugger ★
★ N.S. ISE Support ★
★ Librarian ★
★ Pascal Cross Compiler ★
★ C Cross Compiler ★

U.S. prices start at $500

SOLUTIONWARE
1283 Mt. View-Alviso Rd.
Suite B
Sunnyvale, Calif. 94089
408/745-7818 ★ TLX 4994264

Circle **no. 118** on reader service card.

C Chest Listing (Listing continued, text begins on page 96)

```
{
        /*      Actually do the make. The dependancy tree is descended
         *      recursively and if required, the dependancies are
         *      adjusted. Return 1 if anything was done, 0 otherwise
         */

        TNODE       *snode;             /* Source file node pointer     */
        TNODE       *dnode;             /* Dependant file node pointer  */
        int         doaction = 0 ;      /* If true do the action        */
        static char *zero    = (char *)0;
        char        **linev  = &zero ;


#ifdef DEBUG
        static  int     recurlev = 0;   /* Recursion level              */

        printf("make (lev %d): making <%s>\n", recurlev, what );
#endif

        if( !(snode = find(what, Root)) )
                serr("Don't know how to make <%s>\n", what );


        if( !*(linev = snode->depends_on))  /* If no dependancys        */
                doaction++;                 /* always do the action.    */

        for( ; *linev ; linev++ )           /* Process each dependancy  */
        {
#ifdef DEBUG
                recurlev++;
#endif
                make( *linev );
#ifdef DEBUG
                recurlev--;
#endif

                if( !(dnode = find(*linev, Root)) )
                        serr("Don't know how to make <%s>\n", *linev );

#ifdef DEBUG
                printf("make (lev %d): source    file ", recurlev);
                ptime( what, snode->time );
                printf("make (lev %d): dependant file ", recurlev);
                ptime( *linev, dnode->time );
#endif
                if( snode->time <= dnode->time )
                {
                        /* If source node is older than (time is less than)
                         * dependant node, do something. If the times are
                         * equal, assume that neither file exists but that
                         * the action will create them, and do the action.
                         */
#ifdef DEBUG
                        printf("make (lev %d): %s older than %s\n",
                        recurlev, what, *linev );
#endif
                        doaction++;
                }
#ifdef DEBUG
                else
                    printf("make (lev %d): %s younger than %s\n",
                                                recurlev, what, *linev );
#endif
        }
```

```
358:          if( doaction )
359:          {
360:  #ifdef DEBUG
361:                  printf("make (lev %d): doing action:\n",
362:                                                  recurlev, *linev, what);
363:  #endif
364:                  for( linev = snode->do_this; *linev; linev++ )
365:                  {
366:                          printf("%s\n", *linev); /* Echo action to screen */
367:  #ifndef DEBUG
368:                          if( system(*linev) )
369:                                  serr("Can't process <%s>\n", *linev );
370:  #endif

371:                          /*      Change the source file's time to reflect
372:                           *      any modification.
373:                           */

374:                          snode->time = gtime( snode->being_made );
375:                  }
376:          }

377:  #ifdef DEBUG
378:          printf("make (lev %d): exiting\n", recurlev );
379:  #endif
380:          return doaction;
381:  }


382:  /*------------------------------------------------------------------------
383:   *      Tree routines:
384:   */

385:  TNODE   *find( key, root )
386:  char    *key;
387:  TNODE   *root;
388:  {
389:          /* If key is in the tree pointed to by root, return a pointer
390:           * to it, else return 0.
391:           */
```

(Continued on next page)

C Users' Group

Over 45 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group

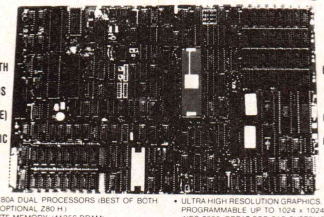
415 E. Euclid • Box 97A
McPherson, KS 67460
(316) 241-1065

Circle **no. 17** on reader service card.

FOR THE **BEST** OF US . . .

THE CYPHER™

A COMPLETE 68000 & Z80A SINGLE BOARD COMPUTER SYSTEM WITH ULTRA-HIGH-RES GRAPHICS!!



FREE 68000 FORTH AND CYPHER-DOS (CPM-80 COMPATIBLE) AND 68000 BASIC

FREE 68000 FORTH AND CYPHER-DOS (CPM-80 COMPATIBLE) AND 68000 BASIC

- 68000 & Z80A DUAL PROCESSORS (BEST OF BOTH WORLDS! OPTIONAL Z80 HI)
- 1 MEGABYTE MEMORY (41256 DRAM)
- DOUBLE DENSITY FLOPPY DISK CONTROLLER (8", 5¼", OR 3½" WD 2793)
- DMA CONTROLLER FOR FAST IMAGE TRANSFERS TO/FROM VIDEO MEMORY. (INT 8237)
- 2 RS232 SERIAL PORTS (Z-SIO)
- 24 BIT ADDRESS MANAGEMENT FOR Z80.
- 4 LAYER P.C.B. (9"x 14"+)
- RUNS CP/M-80 2.2, CP/M-80 3.0, CP/M-68K, CYPHER DOS, RAM DISK, 68000 BASIC IN ROM, NEC 7220 TERMINAL EMULATION.
- ULTRA HIGH RESOLUTION GRAPHICS, PRIVATE 128K PROGRAMMABLE UP TO 1024 x 1024 RESOLUTION (NEC 7220, GREAT FOR CAD SYSTEMS!)
- REAL TIME CLOCK (MULTITASKING CAPABILITY!)
- TWO CHANNELS OF D/A AND A/D, 12 BIT RESOLUTION (OPTIONAL MUSIC! ROBOTICS! LAB WORK!)
- 16K MONITOR EPROM. (EXPANDABLE TO 64K)
- 4K STATIC RAM. (EXPANDABLE TO 64K)
- PROGRAMMABLE BAUD RATE GENERATOR. (8253)
- PARALLEL ASCII KEYBOARD INPUT.
- FULL 68000 EXPANSION BUS. (60 PIN HEADER BUFFERED BUS)

LOWER PRICES! NOW 1 MEGABYTE CYPHER AT $1,299.95

- MANUAL AND COMPLETE SCHEMATICS $20.00
- BARE BOARD, EPROMS, Z80 BIOS, 68000 BIOS, Z80 MONITOR, 68000 MONITOR, CYPHER DOS AND MUCH MUCH MORE $399.95
- KEYBOARD $94.95
- SWITCHING POWER SUPPLY $154.95
- CASE $94.95
- HARD DISC INTERFACE PLUG-IN CARD $150.00
- A/D D/A CYPHER OPTION $100.00

- COMPLETE "CYPHER" WITH 68000, Z80, 1 MEGABYTE DRAM, 128 VIDEO DRAM, NEC 7220, REAL TIME CLOCK, DISC CONTROLLER AND SERIAL I/O, ASSEMBLED AND TESTED $1,299.95

DEALER AND MANUFACTURING FRANCHISE INQUIRIES WELCOME.

SHIPPING CHARGES:
ALL PRICES ARE IN U.S. DOLLARS.
SHIPPING: 8% SHIPPING, EXCESS WILL BE REFUNDED
PRICES SUBJECT TO CHANGE WITHOUT NOTICE
CP/M IS A TRADEMARK OF DIGITAL RESEARCH

MOTOROLA INTEL

MOTEL COMPUTERS LIMITED
174 BETTY ANN DRIVE, WILLOWDALE, TORONTO, ONTARIO, CANADA M2N 1X6
(416) 229-4727

Circle **no. 83** on reader service card.

TURBO PASCAL™ TOOLS

With Super Tools™ and Turbo, YOU can write programs that "pop up" from within Lotus 1-2-3™, dBASE™ etc. In addition, we've included source code to Super Macs, a resident keyboard macro processor!

Routines included let you . .

- Write Sidekick™-like resident programs.
- Redefine keyboard.
- Save and restore screen windows
- Write strings directly to video
- Read strings from screen.

Super Macs™ keyboard enhancer's features:

- 200 simultaneous keys defined.
- 1000 characters maximum per macro.
- Resident macro editor.
- Load & save from within programs.
- Predefined macros for Turbo and Dos.

Additional routines:

- Time and date access.
- Dos program execution and return.
- Dos memory allocation.

For your copy of Super Tools send $54.95 to:

Sunny Hill Software
13732 Midvale N., Suit 206 • Seattle, WA 98133
(206) 367-0650

Turbo Pascal & Sidekick trademark Borland Int'l. Lotus 1-2-3 Reg. trademark Lotus Dev. Corp. dBase trademark Ashton Tate.

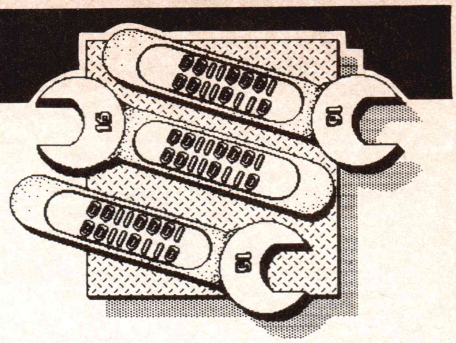
Circle **no. 10** on reader service card.

```
392:        register int    notequal ;
393:        register TNODE  *rval    ;

394:        if( !root )
395:                return 0;

396:        if( !(notequal = strcmp(root->being_made,key)) )
397:                return( root );

398:        return(  find( key, (notequal > 0) ? root->lnode : root->rnode) );
399: }

400: /*---------------------------------------------------------------------------*/

401: tree( node, rootp )
402: TNODE   *node, **rootp ;
403: {
404:        /* If node's key is in the tree pointed to by rootp, return 0
405:         * else put it into the tree and return 1.
406:         */

407:        register int    notequal ;
408:        register TNODE  *rval    ;

409:        if( *rootp == NULL )

410:        {
411:                *rootp = node;
412:                return 1;
413:        }

414:        if( !(notequal = strcmp( (*rootp)->being_made, node->being_made)))
415:                return 0;

416:        return( tree( node, notequal > 0 ? &(*rootp)->lnode
417:                                         : &(*rootp)->rnode)   );
418: }

419: /*---------------------------------------------------------------------------*/

420: main( argc, argv )
421: char    **argv;
422: {
423:        /*      A stupid version of the unix make utility.
424:         *
425:         *
426:         */

427:        if( !(Makefile = fopen(MAKEFILE, "r")) )
428:                err("can't open %s\n", MAKEFILE );

429:        if( !dependancies() )
430:                err("Nothing to make");
431:        else
432:                make( argc > 1 ? argv[1] : First );
433: }
434: #ifdef DEBUG
435: /*---------------------------------------------------------------------------
436:  *      Msc. Debugging routines
437:  */

438: ptime( file, t )
439: char    *file;
440: long    t;
441: {
```

```
442:          /* Print out the time and date field of a TNODE as
443:           *              "mm-dd-yy hh:mm:ss"
444:           * File is the file name.
445:           */

446:          int date, time;

447:          date = (t >> 16) & 0xffffL ;
448:          time = t & 0xffffL ;

449:          printf("%s: ", file );
450:          printf("%02d-%02d-%02d, ", (date >> 5 ) & 0x0f, date & 0x1f,
451:                                          80 + ((date >> 9) & 0x7f) );

452:          printf("%02d:%02d:%02d, ", (time >> 11) & 0x1f, (time >> 5) & 0x3f,
453:                                            (time << 1) & 0x3e  );
454:          printf("\n");
455: }

456: /*-------------------------------------------------------------------------*/

457: pnode( node )
458: TNODE *node;
459: {
460:          /*      Print out the tree node pointed to by "node"
461:           */

462:          char **linev;
463:          printf("+----------------------------\n");
464:          printf("|  node at 0x%x\n",                          node );
465:          printf("+----------------------------\n");
466:          printf("| lnode  = 0x%x,  rnode = 0x%x\n",node->lnode,node->rnode);
467:          printf("| time   = 0x%lx =",              , node->time          );
468:          ptime( "", node->time );
469:          printf("| target = <%s>\n"                , node->being_made    );
470:          printf("| dependancys:\n"                                       );
471:          for( linev = node->depends_on; *linev; printf("|\t%s\n", *linev++))
472:                  ;

473:          printf("| actions:\n" );
474:          for( linev = node->do_this; *linev; printf("|\t%s\n", *linev++))
475:                  ;
476:          printf("+----------------------------\n");
477: }

478: /*-------------------------------------------------------------------------*/

479: trav( root )
480: TNODE   *root;
481: {
482:          /*      Do an in-order traversal of the tree, printing the
483:           *      node's contents as you go.
484:           */

485:          if( root == NULL )
486:                  return;

487:          trav( root->lnode );
488:          pnode( root );
489:          trav( root->rnode );
490: }
491: #endif
```

End Listing

16-BIT SOFTWARE TOOLBOX



by Ray Duncan

Bug Report: TEE Filter

Dr. Fred Sinal of Hampton, Virginia, writes: "There is a slight bug in your listing of TEE in the April 1985 '16-Bit Software Toolbox' column. Actually the bug is in DOS, but that's life. If the file does not end with a CR/LF, we get a 'disk is full' message. DOS is forcing CR/LF to the console and confusing the character count. This problem may be covered up by eliminating the pair of lines

```
cmp ax,nchar
jne tee7
```

before the label **tee4**. Of course this reopens the disk full problem, when redirecting the standard output. I could not find a truly satisfactory fix."

Some Musings about DRI

Most of you have probably seen the recent splashy full-page ads for Digital Research's GEM that are appearing in nearly every magazine except *Sports Illustrated*. These ads lead you to believe that you can get a 16-color equivalent of Macintosh MacPaint for the IBM PC by just picking up GEM and GEM-Paint for a few pence at your corner computer store.

Beware: The fine print in the ad says "appropriate graphics hardware required." The appropriate hardware turns out to be the IBM Enhanced Graphics Adaptor, which will set you back about $1200 for the required controller board, memory expansion, and higher resolution RGB monitor. If you use GEM on an "ordinary" IBM graphics adaptor, the icons, windows, and all the rest are in vibrant black and white.

Speaking of Digital Research, look before you leap into spending any money on the product that DRI is advertising (with tongue in cheek, I'm sure) as "Concurrent PCDOS." It's concurrent, I guess, but it certainly doesn't resemble current versions of PCDOS. "Concurrent PCDOS" does not support the hierarchical file structure or any of the Unix-like executive services that are present in MSDOS or PCDOS versions 2 and 3. In other words, "Concurrent PCDOS" supports only the functionality of MSDOS or PCDOS version 1, which was more or less a clone of CP/M.

Thus, almost any of the more powerful MSDOS applications you are likely to buy will not run properly under "Concurrent PCDOS." Furthermore, a recent review in *PC Tech Journal* demonstrated that a set of programs run concurrently under "Concurrent PCDOS" will actually take longer to complete than if you simply run them consecutively under normal MSDOS. You can draw your own conclusions, but Digital Research's hopes of recapturing its old dominance of the personal computer operating system market seem more like delusions to me.

More on Square Roots

Tom Prince, of Marblehead, Massachusetts, writes:

"I was interested in your discussion of square rooting in the May 1985 column. A similar algorithm in 8088 code appeared in *Dr. Dobb's* over a year ago, encouraging me to add a floating point square root to my collection of Z80 code. Since the Z80 has enough registers to perform 32-bit calculations, it can do the square root much faster than the 8080 or even 8088 can do a floating point divide. Imagine my surprise to see an integer square root that takes ten times as long as a divide, even when coded in assembler! Even a conversion to floating point to take the root ought to be faster.

"So I wrote up the following code for a well-known '32-bit' processor. It turns out to be slower than sqrt(float( )) but certainly doesn't take ten times as long as an integer divide. Of course it takes longer than the three integer divides that are included and would run faster if coded in assembler. Would it be too much to ask to provide high-level language versions of routines like these that work perfectly well, if a bit slower?"

Mr. Prince's code can be found in Listing One (page 122).

Mac Fan Strikes Back

Mr. S. Allen, of Ojai, California, writes:

"I am, and have been for quite a while, an ardent admirer of *DDJ*. I have been hard-pressed to find a magazine of such technical excellence aimed at the more advanced computer user.

"I do, however, have a GRIPE! I was quite excited when you began the 16-Bit Toolbox. I expected to find droves of information on the latest hardware innovations and the support software to go with them. You can imagine my chagrin when I discovered a thinly disguised 'The IBM PC is Wonderful' column. To call the IBM PC, with its 8088, a 16-bit machine is pushing things a little, and then to have the writer launch irrational tirades against a true 16-bit machine (the Mac) lit my Bulls—t High Level Light.

"Please, be honest. If you are going to have an IBM PC column, call it that. If you are going to have a 16-bit column, let's have it be about 16-bit machines. From reading the present 16-Bit Toolbox, I must assume that the writer is either in IBM's or Intel's pocket. Your magazine has a long-standing reputation for impartiality. I implore you, do not destroy it."

Well, I suspect that the flip-flop

controlling Mr. Allen's Bulls—t High Level Light is in need of replacement. Let me first assure *DDJ* loyal readers that I am not in either IBM or Intel's pocket. If I were, I'd ask them for a suitably monstrous amount of money, retire, and pass this column on to other younger and more energetic souls.

People who actually read this column know that it is heavily devoted to reports of bugs in Microsoft, IBM, or Intel products, so that programmers can work around them. These bug reports hardly constitute advertisements for the companies involved (except, I guess, for those who have the philosophy that any publicity is good publicity). I defy anyone to look back at the previous columns and anywhere find the message "The IBM PC is Wonderful."

As for our emphasis on the Intel 8086 family of microprocessors and the MSDOS operating system, these remain the dominant force in the personal computer world and seem likely to remain so for the forseeable future; consequently, they are of primary interest to those of us who write programs for a living. Whether the 8088 is really a 16-bit processor or not is a moot point; it runs the same machine code as the 8086, it acts like an 8086, and it is everywhere. Quibbling over the bus width is a waste of our time.

We certainly are interested in extending our coverage of other 16-bit processors, and as people who actually read this column will also know, we have given considerable coverage to the 68000 and Macintosh over the last few months and will continue to do so as long as our readers indicate an interest in it.

Regarding 68000 Assemblers

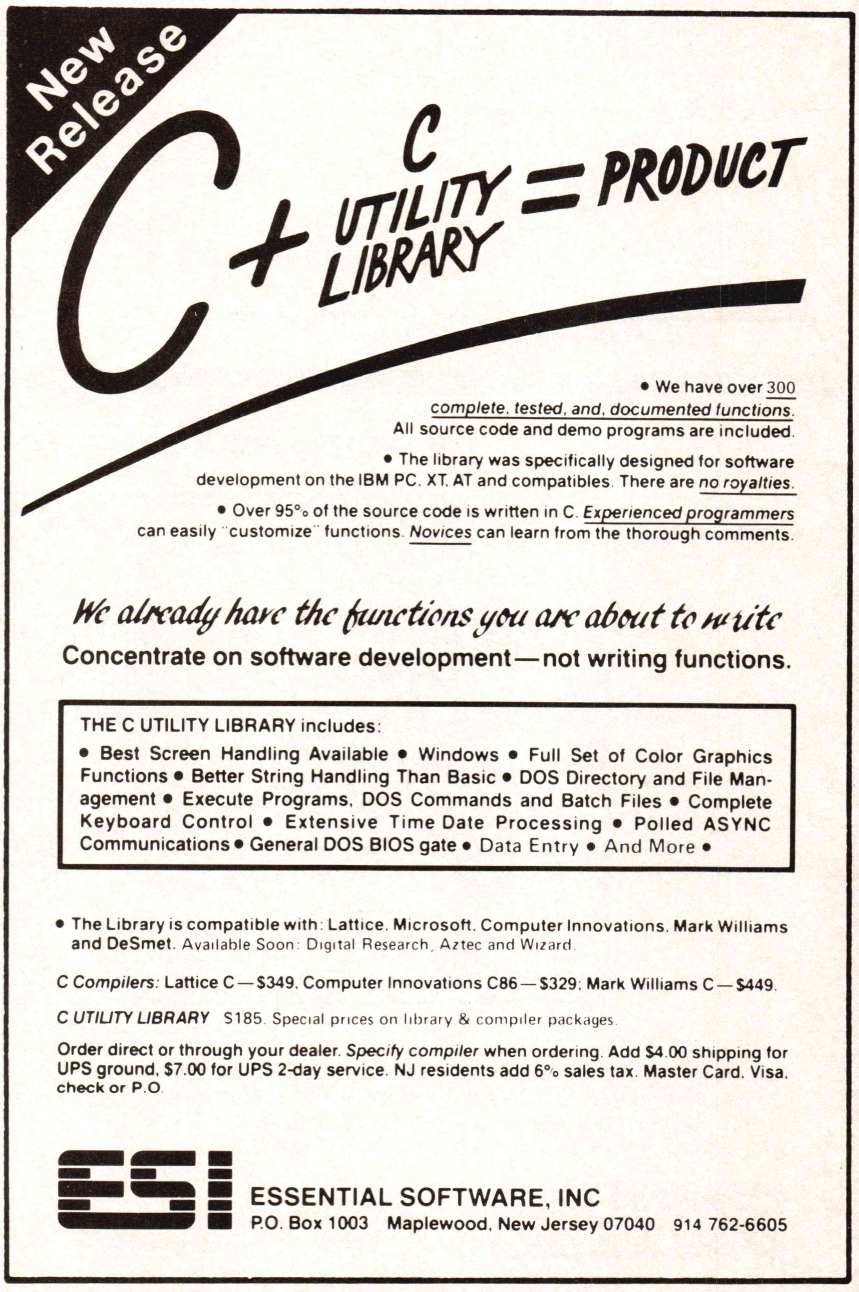
Mr. Michael Aichlmayr, of Tacoma, Washington, writes:

"In regards to Mr. Howell's remarks in your May 1985 column, I would like to add myself to the list of people who feel it is truly possible to build an efficient assembler for the 68000—one that would run in a reasonable amount of memory and execute in a productive amount of time.

"I don't feel that the 68(x)xx family is nearly as complicated as Mr. Howell implies. I think elegant simplicity is a more accurate statement. Any good programmer could make great use of generic addressing modes and instructions. I think I would hazard to go so far as to say that I believe it could be done in C and run in 64K on a 68(x)xx processor (the horrible overhead of the [Intel] 80(x)xx would make it a real challenge). I have a 6809 assembler written in C that runs easily in 64K of memory on every machine for which it is compiled.

"It's very sad to realize that the abundance and low cost of memory have given programmers an excuse to be gross and inefficient. I have seen code that could run in my HP 29C calculator take up more than 64K of memory and 'require' a 128K IBM PC or compatible to run. It makes me think that if a machine came standard with a megabyte of RAM, someone would come up with a desk calculator that would hog the machine.

"Mr. Howell remarks that he has a 6809 system that he likes better than his IBM compatible. I direct his attention to the TSC (Technical Systems Consultants) 68000 macro assembler. I have been able to coax the assembler to run in 32K on a 6809



Circle no. 36 on reader service card.

edward k. ream

Transform Your Programs with CPP—C Preprocessor Plus

Includes ALL features of the standard C preprocessor.

- Define arbitrarily complex macros with #define command.
- Include and nest files to any depth with #include command.
- Conditionally include or exclude lines with #if, #ifdef and #ifndef commands.
- Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or * are ignored.)

Fast and flexible

- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
- Can be used for any language, including assembler.
- Can be used as a stand-alone macro/include processor.
- Code can be used as the lexical analyzer for parsers or assemblers.

Complete

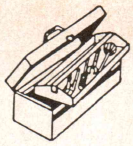
- You get complete SOURCE CODE in standard C.
- You get everything you need to use CPP immediately.
- CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

Price: $50. Call or write today:

Edward K. Ream
1850 Summit Ave., Madison, WI 53705
(608) 231-2952

TO ORDER: Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5¼ inch disk). Send a check or money order for $50 ($60 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, I do NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check in included.

Circle **no. 13** on reader service card.



Where did AT&T and SONY find the Tools to C them thru?

The Application Programmer's Toolkit!!!

A *Wizard!Ware*™ product from Shaw☆American Technologies

APT™ provides you with everything you need to increase your C programming productivity, including:

- COMPLETE SOURCE CODE (over 5000 lines!)
- File handling with direct & keyed access
- Screen and Report Generators, with full screen handling for your programs
- Generic Terminal Driver for portable code
- String math functions, and string manipulation routines
- Reference Manual on Disk (over 50 pages)
- Tutorial Manual (over 25 pages) with Source for Mailing List Manager
- A host of useful Utilities, Database and File Editors
- Available for Microsoft (3.0), Lattice, Mark Williams, DeSmet, BDS, Aztec, CI-C86

C-STARTER Toolkit!: Binary APT, DeSmet C, "Programming in C on the IBM-PC"

NOW MORE AFFORDABLE!!

APT/MS-DOS versions	$395
APT/DeSmet C & BDS C versions	$295
C-Starter (binary APT, DeSmet Compiler and Book)	$295
APT/Manual only	$ 45

**** NEW PRODUCTS! Available Now or Coming Soon: ****

ADAPT: English-language, applications generator	$295
BIZ-WIZ: Comprehensive Accounting Package	$495
FLORA: Graphics Toolkit, with MAC-like capabilities	$ 95
APT-WINDOWS!: APT-Compatible/stand-alone window mgt.!	$ 95
db2c: dBaseII/III TO C Source Code Translator and Libraries!	CALL
Dr. Shaw's DOS-Shell: a UNIX-like shell for DOS	$ 95

Trademarks: MS DOS & Microsoft C/Microsoft, Lattice C/Lattice, Inc., MWC86/Mark Williams Co., DeSmet C/C Ware, CI-C86/Computer Innovations, BDS C/BD Software, Aztec/Manx Software, UNIX, AT&T/AT&T, SONY/Sony Corp., Macintosh/Apple Computer, dBaseII & dBaseIII/Ashton Tate, ADAPT/Wilmes Systems, FLORA, Biz Wiz, WizardWare, APT, C-Starter/Shaw - American Technologies.

Call (502) 583-5527

Shaw☆American Technologies

Wizard Ware™

VISA

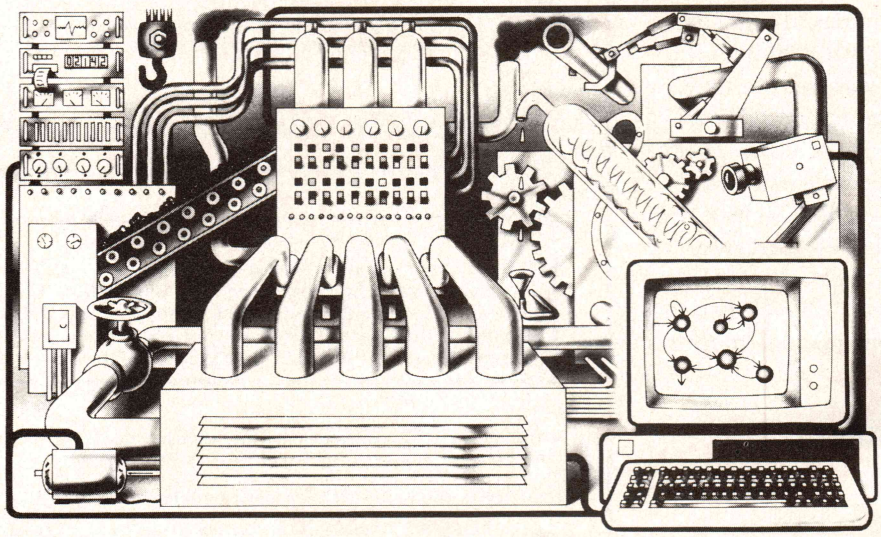
830 South Second St. - Box 648
Louisville, KY 40201, USA

master charge

(C.O.D. and Foreign Orders - Add $5 Shipping/Handling)
References: Bank of Louisville, Citizens Fidelity Bank, Louisville Chamber of Commerce

Circle **no. 86** on reader service card.

Csharp Realtime Toolkit



Realtime on MSDOS? Csharp can do it! Get the tools without operating system overhead. Cut development time with C source code for realtime data acquisition and control. **Csharp** includes: graphics, event handling, procedure scheduling, state system control, and interrupt handling. Processor, device, and operating system independent. **Csharp** runs standalone or with: MSDOS, PCDOS, or RT11. **Csharp** runs on: PDP-11 and IBM PC. **Csharp** includes drivers for Hercules and IBM graphics boards, Data Translation and Metrabyte IO boards, real time clock, and more. Inquire for Victor 9000, Unix, and other systems. Price: $600

SYSTEMS GUILD

Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

Circle **no. 50** on reader service card.

system running under the FLEX operating system. I realize it does not produce relocatable object modules; however, that piece of code would be fairly small given the architecture.

"Having a 6809 machine (which I built), a Mac (which I tolerate), and a DG One IBM compatible (which I use), I think I can safely assume that I have some knowledge of hardware and software. I find it very aggravating that IBM chose the 80(x)xx architecture over that of the 68(x)xx, but I have never seen a processor more crippled than that in the Mac.

"I eagerly await the arrival of the Atari ST, which I have had the opportunity to play with for a while. If all appearances are correct, this is a good use of a 68000. The machine runs at 8 MHz, has 32K [of RAM devoted to] memory-mapped color graphics, and is said to come standard with 512K of RAM for $699.00! I'm not sure what the GEM OS is going to do to the thing since I have little regard for DRI, but it doesn't look bad so far."

More Mac Feedback

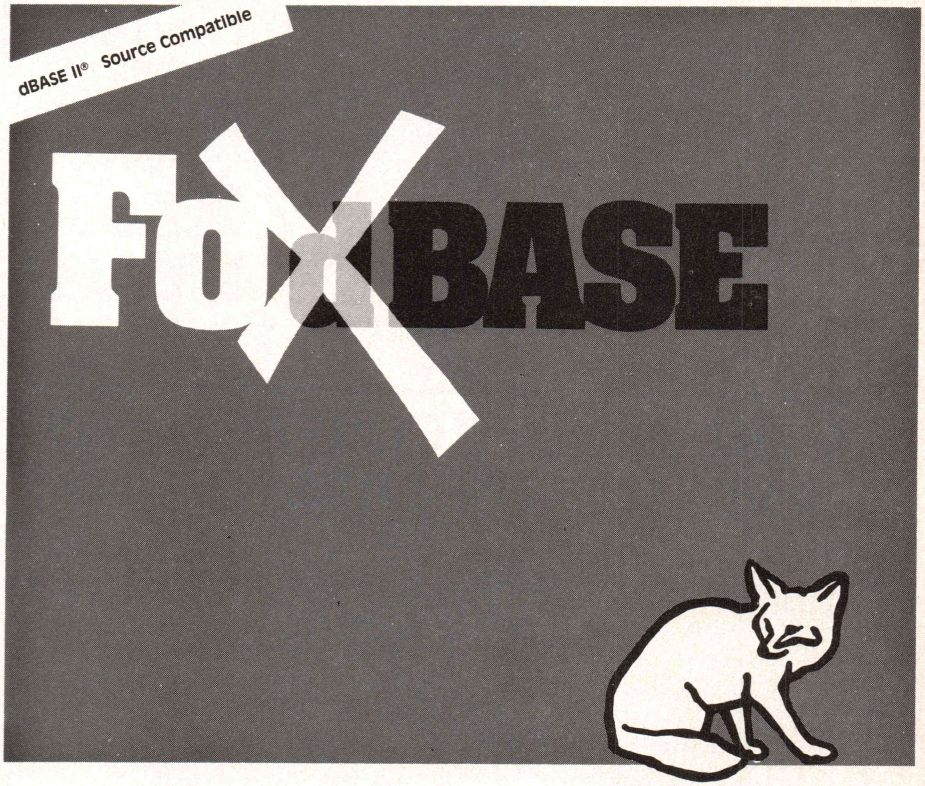
Kirk Kerekes, of Tandata Inc., in Tulsa, Oklahoma, submitted a thoughtful and interesting letter:

"As circumstances seem to have started a debate in your column *re* the Mac as a development environment, I felt it appropriate to provide some facts to fuel the debate and perhaps redirect it to more substantive issues.

"FACT: There is no assembler that I know of for the Mac that requires the use of two (or more!) Macs. This myth has been floating around for quite a while and was recently mentioned in your column. This myth doubtless arises from the fact that there are two Apple-issue debuggers that use a second computer as a control device to allow undisturbed screen displays on the system being debugged. As there are around half a dozen debuggers that do *not* require a second unit, and at least one of the two-machine debuggers will use *any* reasonable terminal as a remote controller, I see no reason to regard this situation as anything but a blessing for the developer. The debuggers are available as part of the Inside Mac supplement program and are distributed as a group.

"If you are doing development on the Mac and are a fan of debuggers, then the $100 price tag of the supplement could be justified for them alone. (I've gotten at least $100 worth of micro-floppies through my supplement subscription, just valued as media!)

"FACT: There are numerous complete, stable high-level languages for the Mac. A recent glance reveals at least three Forths, half a dozen Cs, a Pascal or two, Microsoft BASIC, and NEON, an object-oriented Forth-like language. I can state that at least one product of a professional nature, MegaMax C, is 'real,' well-supported, and quite clean. Assembler fans can obtain the MacAsm mentioned in your column or ask almost anybody for a copy of the prerelease versions of the Apple Editor/Assembler/Linker/Resource Compiler package. By the time this might see print, the official release of the MDS+Inside Mac package should have occurred. This package is already on the certified developer's price list. The price is lower than any complete serious development environment available for



Evolution.

Now FoxBASE, the dBASE II source-compatible interpreter/compiler, is even better than before. Automatic 8087 co-processor support allows you to perform numeric computations with lightning speed. Fourteen-digit precision gives you 40% greater accuracy than dBASE II. The fact that FoxBASE is not copy protected means you can easily load it onto your hard disk. What's more, FoxBASE comes complete with a NO-RISK demo plan.

Of course, FoxBASE still offers all of the features that dBASE II does . . . PLUS • Runs 3 to 20 times faster • Permits up to 48 fields/record...50% more than dBASE II • Supports full type-ahead • Compiles program sources into compact object code • Has twice as many variables • Comes with a sophisticated online manual and HELP facility.

FoxBASE is currently available on a wide range of machines: IBM-PC, IBM-PC/XT/AT, COMPAQ & IBM compatibles, TI Professional, DG Desktop, and DG MV-Series to name just a few. And it will soon be available on the Molecular and NCR Tower computers as well. Call or write today for more information.

MS-DOS:	Development Pkg.	$395
	Runtime Pkg.	$695
AOS/VS:	Development Pkg.	$995
	Runtime Pkg.	$1995
UNIX and XENIX:	(To Be Announced)	

Developed by DACOR COMPUTER SYSTEMS

dBASE II is a trademark of Ashton-Tate. UNIX is a trademark of AT & T. FoxBASE is a trademark of Fox Software Inc.

FoxBASE™ from FOX SOFTWARE INC.

13330 Bishop Road, P.O. Box 269, Bowling Green, OH 43402 / 419-354-3981 / TWX 810-499-2989

Circle no. 40 on reader service card.

MSDOS.

"FACT: The bloated size of most of the early Mac applications is the result of the use of Lisa Pascal as the development environment. There is absolutely nothing about the Mac that requires/generates bulky code. I have created useful Mac-interface programs with a load module size less than 1K using the MegaMax C compiler. An 'empty' C program (i.e., main( ) {}) yields a load module of about 300 bytes. Most of this appears to be information that any application stores for use by the Finder. I just tried the same experiment with Computer Innovations C86 under MSDOS and got a 5K EXE file result, due to C86's insistence on loading the standard I/O library whether or not it is invoked.

"POINT: Nobody at Apple denies that the Mac concept, from mouse on up, is derived from the Xerox PARC research. The point in the Mac is to take the PARC concepts and implement them in an affordable machine. Go price a Xerox Star and then price a Mac (first, *find* a Star!)." [As this column went to press, Xerox announced a new incarnation of the Star, the model 6085, that costs $4995 with 1 Mb of RAM, 10 Mb hard disk, a 15-inch display, and a mouse . . . . RD]

"OPINION: I develop professionally on MSDOS, CP/M, and Mac systems. I use the major development products in each environment for both C and assembly language. It is my opinion that there are better development environments available for the Mac than for MSDOS or CP/M. The Mac development products tend to be less expensive, too. There are also some real losers available for all these environments. I'm not going to name names; we all have our own favorite turkeys.

"OPINION: There are a number of programming professionals who feel threatened by the Mac. It is not my purpose to speculate why this should be the case. It *is* my purpose to assert that no professional need feel threatened by the Mac. With the tools currently available, it is a joy to develop on, and there are enough 'goodies' in the system toolbox to keep just about anybody happy. There is definitely a learning curve involved, enough of one to be a barrier to dabblers, but anyone familiar with good modern programming practices can produce useful programs in very short order. The Mac environment punishes sloppy programming practices and greatly rewards programmers who plan thoroughly before they code.

"Maligning the Mac is not going to make it go away. Being annoyed by Steve Jobs or marketing hype is not realistic. The only tool Apple has to battle IBM in the public mind is flamboyance. Steve Jobs' opinions may not be yours, but you know his name, *Playboy* readers know his name, and an amazingly large percentage of the general public knows his name. Now, what is the name of the CEO of IBM? Of *any* officer of IBM? See how it works?

"I propose that further discussion of development on the Mac actually center on development on the Mac . . . not on myths and spurious side issues! There is plenty to discuss, believe me!"

In reply, I only question whether the only tool Apple has to battle IBM is flamboyance. The tried and true tools of delivering quality products on time seem to have served some companies pretty well (such as Compaq, which has taken on IBM head-on and is doing better than Apple at it in many respects). The rest of Mr. Kerekes' points are well taken, and indeed we will try to stick to substantive discussion of the Mac in the future. By the way, the CEO of IBM is John Akers (last I heard); although this fact probably *isn't* a part of the general knowledge of the average *Playboy* reader, it is fairly common knowledge in the computing community.

Reading about the Mac

I have run across two recent articles about the Macintosh that are both helpful and balanced (unlike the rabid pro or con articles that are prevalent in the popular computer press).

9 TRACK TAPE CONTROLLERS AND ½" TAPE SUBSYSTEMS

MODEL TC-PC

TC-PC is a high performance 9-track tape controller for the IBM-PC with these important features:

- Reads and writes industry standard ½-inch tape
- Compatible with most formatted tape drives
- Standard 8-bit parallel recording with parity, and read-after-write verification
- Switch selectable I/O address (four contiguous ports required for operation)
- Maximum data transfer rate of 192,000 bytes per second
- Record length from 1 to 65,535 bytes
- Supports up to 8 tape transports
- Jumper selectable DMA channel
- Modes: PE and NRZI at 800, 1600, 3200 and 6250 bytes/inch
- Installable device drivers allow creation of application programs which run under IBM XENIX and MS-DOS
- Operates with IBM-PC and -XT; Compaq Portable; Zenith PC-150; Sperry PC; the Leading Edge Computer, and other 100% IBM-PC compatible equipment.

MODEL TC-50

TC-50 offers all the standard features of the TC-PC with these additional enhancements:

- Maximum data transfer rate of 400,000 bytes/second; 904,000 bytes/second with memory option
- Operation with a wider range of IBM-compatible machines, including IBM-AT; Compaq Desk Pro; ATT 6300 and others

A variety of software utilities is supplied as part of the TC-PC and TC-50 packages, including:

- **DEPOT (Data Exchange Program with Optional Translation)**
  DEPOT provides a means to transfer data between system disk and magnetic tape, allowing:
  — Data interchange from tape to disk, and disk to tape
  — Conversion from ASCII to EBCDIC, and vice versa
  — Positioning to arbitrary location prior to data read
  — Specification of record length and block factor when writing from disk to tape; allows deblocking when reading from tape to disk
  — Multiple operations to be specified from a command file
- **TAU (Tape Archive Utility)**
  — Provides individual file backup and restore
  — Allows use of MS-DOS wild cards such as "*.*"
  — Provides disk drive selections for I/O
  — Changes pathname selections from within TAU
  — Provides data encryption for security

WARRANTY

All Overland Data products carry a 30-day unconditional money-back guarantee, and are warranted for one year, parts and labor.

XENIX and MS-DOS are Registered Trademarks of Microsoft Corp.
IBM-PC/AT/XT are Registered Trademarks of International Business Machines Corp.
Compaq Portable and Compaq Deskpro are Registered Trademarks of Compaq Corp.
AT&T 6300 - AT&T Information Systems Corp.
Sperry PC - Sperry/Univac Corp.
Zenith PC-150 - Zenith Data Systems
Leading Edge is a Registered Trademark of Leading Edge Products, Inc.

OVERLAND DATA, INC.
5644 Kearny Mesa Road #A
San Diego, CA 92111
Tel. (619) 571-5555

Circle **no. 39** on reader service card.

DSD 80

FULL SCREEN SYMBOLIC DEBUGGER

"THE SINGLE BEST DEBUGGER FOR CP/M-80. A TRULY AMAZING PRODUCT."

LEOR ZOLMAN
AUTHOR OF BDS C

- ☐ Complete upward compatibility with DDT
- ☐ Simultaneous instruction, register, stack & memory displays
- ☐ Software In-Circuit-Emulator provides write protected memory, execute only code and stack protection.
- ☐ Full Z80 support with Intel or Zilog Mnemonics
- ☐ Thirty day money back guarantee
- ☐ On-line help & 50 page user manual

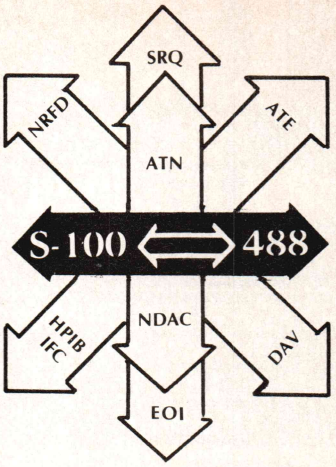
NOW ONLY $125.

SOFTADVANCES

P.O. BOX 49473 AUSTIN, TEXAS 78765 (512) 478-4763

MasterCard VISA

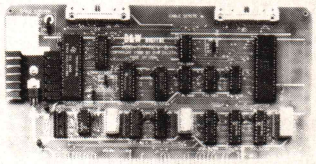
Circle **no. 63** on reader service card.



IEEE 488 TO S-100 INTERFACE

- Controls IEEE 488 (HPIB) Instruments with an S-100 computer
- Acts as controller or device
- Basic and assembly language drivers supplied
- Meets IEEE 696 specification
- Industrial quality burned in and tested up to 125K bytes/sec under software control 3 parallel ports (8255-5)
- $375

THE 488+3



D&W DIGITAL, INC.
20655 Hathaway Avenue
Hayward, California 94541
(415) 887-5711

Circle **no. 28** on reader service card.

THE BEST Z80 ASSEMBLER ON THE MARKET JUST GOT BETTER!

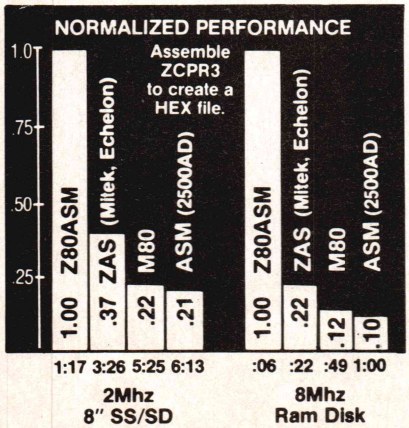
Z80ASM

NOW ONLY **$49.95**

DON'T ASK HOW OURS CAN BE SO FAST . . . ASK WHY THEIRS ARE SO SLOW!

". . a breath of fresh air . ."
Computer Language, Feb. 85

". . in two words, I'd say speed & flexibility",
Edward Joyce, User's Guide #15



Now fully compatible with M80 in .Z80 mode with many extensions. Time & date in listing, 16 char. externals, plus many other features.

To order, or to find out more about our complete family of development tools, call or write:

SLR Systems

1622 N. Main St., Butler, PA 16001
(800) 833-3061, (412) 282-0864
Telex 559215 SLR SYS

MasterCard VISA C.O.D., Check or Money Order Accepted

SHIPPING: USA/CANADA + $3 • OTHER AREAS + $10
Z80 CP/M compatibility required.

Circle **no. 78** on reader service card.

LISP
FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE OF ARTIFICIAL INTELLIGENCE FOR YOUR IBM PC.

- **DATA TYPES**
  Lists and Symbols
  Unlimited Precision Integers
  Floating Point Numbers
  Character Strings
  Multidimensional Arrays
  Files
  Machine Language Code
- **MEMORY MANAGEMENT**
  Full Memory Space Supported
  Dynamic Allocation
  Compacting Garbage Collector
- **FUNCTION TYPES**
  EXPR/FEXPR/MACRO
  Machine Language Primitives
  Over 190 Primitive Functions
- **IO SUPPORT**
  Multiple Display Windows
  Cursor Control
  All Function Keys Supported
  Read and Splice Macros
  Disk Files
- **POWERFUL ERROR RECOVERY**
- **8087 SUPPORT**
- **COLOR GRAPHICS**
- **LISP LIBRARY**
  Structured Programming Macros
  Editor and Formatter
  Package Support
  Debugging Functions
  .OBJ File Loader
- **RUNS UNDER PC-DOS 1.1 or 2.0**

IQLISP

5¼" Diskette and Manual _____ $175.00

∫q *Integral Quality*

P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

The first is simply a detailed, objective, nonhysterical review of the Macintosh written by Larry Press of Small Systems Group; this review appeared in *Abacus*, Vol. 2, No. 2 (Winter 1985), page 51. The author is an experienced computer user with no particular fish to fry who is oriented toward the use of the Macintosh in educational contexts.

Abacus, by the way, is an intriguing publication that shows signs of developing into a unique and useful magazine. The quality of writing is high, the percentage of advertisements low, there is virtually no boosterism of "in" computers or software products, and the magazine is eclectic. The issue mentioned contained articles on human computers (how calculating prodigies perform arithmetic), a report on personal computers at Amos Tuck School, an exposition of "The Applicative Style of Programming," a review of the special *Scientific American* issue on software, and a report by chess master David Levy on a match between himself and CRAY BLITZ (winner of the 1983 World Computer Chess Championship). *Abacus* is published by Springer Verlag New York Inc., a source of scientific and academic texts.

The second notable article is "Apple Macintosh Software" by Dr. Ted Lewis of the Computer Science Department of Oregon State University, published in *IEEE Software*, March 1985, page 89. This is a combination review and tutorial of "the coding standards and methods forced upon application programmers who accept the challenge of writing applications for the Macintosh." It includes a simple example program with pull-down menus and an explanation of the structure and contents of the resulting "resource file." This article is the first reasonable explanation I have seen of this subject and is orders of magnitude more comprehensible than the Apple documentation.

Dr. Lewis concludes: "The ideas and techniques incorporated into the Macintosh software are admirable, but I am struck by the boldness of Apple Computer. Programming a Macintosh is not easy for conventional programmers, and this has contributed to the slow appearance of Macintosh software. Does Apple expect everyone to change? Do they think that a technical achievement like the Macintosh is inducement enough to persuade programmers to wade through 800 pages of *Inside Macintosh* merely to discover the merits of object-oriented programming and Desktop?

"The Macintosh software is unconventional, nonstandard, and probably horribly nonportable. The dialect of Pascal used is related to UCSD Pascal, but there is little hope of separating it from the Macintosh ROM routines. Yet, the software is nearly a work of art; all objects are organized in a logical and concise manner. The notion of a resource file is powerful and will most likely be emulated by programmers everywhere."

DDJ

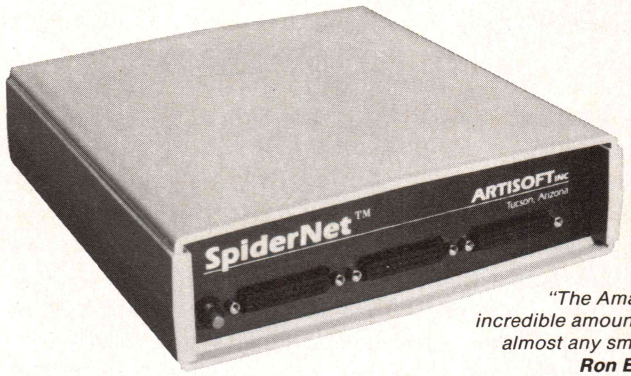
Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 196.**

```
function isqrt(i)
                integer*4 i,js #32 bits
                byte j1(4)
                equivalence(js,j1(1)) #with reversal, j1(4) is high byte of js
#you didn't take care of zero and negative arguments!
                js=i
                #count high order 0 bits 8 at a time
                for(k0=0;j1(4)==0;k0=k0+8)js=ishft(js,8) #shift left 8 bits
                for(;js>0;k0=k0+1)js=ishft(js,1) #count remaining 0 bits
                #get initial approximation within 50%
                isqrt=ishft(3,ishft(31-k0,-1)-1) #3*2**((29-k0)/2),not for i=1
                do k=1,3 #improve by Newton iterations
                        isqrt=ishft(i/isqrt+isqrt,-1) #(i/isqrt+isqrt)/2
return
end
```

Listing One
Tom Prince's square root algorithm

FINALLY!

Communications Hardware and Software that's Inexpensive, Easy to Use and Guaranteed.



"The Amazing SpiderNet . . . an incredible amount of utility available to almost any small-sized application."
Ron Exner, 12/84 Hardcopy

SNARE™—SpiderNet's Callback Security System — $595

- Protects ANY dialup computer system from unauthorized intruders
- Authorized users are called back at specific locations
- SNARE secures three modem lines at less than $200/line
- Up to three separate computers can use SNARE simultaneously
- Uses Hayes-compatible modems or others like DEC's DF03
- Stores 70 authorized users, expands to 150 (300+ soon)

SpiderNet—Printer/Plotter Sharing Unit — $495

- Shares expensive RS-232 plotters, daisy wheel or laser printers
- Makes laser printers affordable for your office or department
- Two to five micros or minis can share a peripheral
- Connect and use—no software modifications required!
- Optional 64K buffer for spooling

SpiderNet—Computer and Peripheral Networking — $495

- Six port, intelligent software-controlled RS-232 switch
- Share multiple peripherals between systems
- Interconnect computers to share data and files
- Three pairs of ports can be connected simultaneously
- Links ports at different baud rates
- Programmable: perfect for custom RS-232 control application

SpiderNet—Multiplexer, Terminal Concentrator and more — $495

- Five to one RS-232 multiplexer/demultiplexer
- 5×1 or 4×2 concentrator for expanding terminal ports

ENVOY™—Telecommunications Software — $49.95

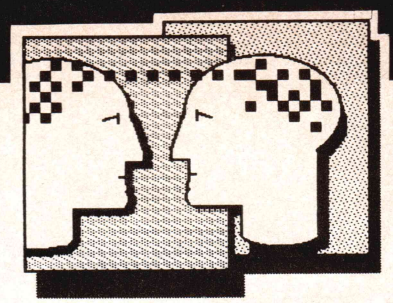
- Access to electronic mail, remote systems and data networks
- Error free, text and binary file transfers via XMODEM or ANSI X3.28
- Smart terminal mode with capture buffer, autodial and autologin
- Easy to use, menu driven, compact and high speed
- Utilities menu for copy, type, print, erase and rename files
- For IBM PC, PCjr, PC compatibles, Sanyo MBC-55X, CP/M-80 or -86

30-day money-back guarantee on all products

ARTISOFT INC
Box 41436, Tucson, Arizona 85717
(602) 327-4305

Circle **no. 51** on reader service card.

CP/M EXCHANGE



by Robert Blum

The CP/M Exchange RCP/M system is available for your use 24 hours a day, 7 days a week. Reach it by dialing (404) 449-6588.

Online . . . at Last

For more than two years I have been promising to bring up a RCP/M system in support of this column—finally, it's online. I hope you will participate in its use and share with me your thoughts on how I can improve it.

To reach it, call (404) 449-6588 any hour of the day or night, seven days a week. The modem is a Racal-Vadic 3452, which is capable of communicating at either 300 or 1200 baud. The other hardware components are an Ampro single-board Z80 microcomputer driving two 5¼-inch Teac floppy disk drives and a 20 Mb hard disk drive. This configuration should be more than adequate, at least for the present, to support the amount of activity that I anticipate.

The main purpose of the system is to provide improved access to the information printed in this column. The complete text of all the programs and updates printed here will be available for downloading. I hope this will stimulate even more reader participation in the column by providing a convenient method for submitting items of interest. I will also maintain a current list of all known CP/M bug reports and any fixes that might be available; this includes application notes and fixes published by DRI.

When your call is answered, hit the return key several times to indicate the baud rate you are using. Next you will be asked whether your terminal needs nulls for proper operation. Once beyond these two preparatory steps, you will enter into the message system. From this point on, I think the system is self-prompting enough to get you started without requiring a lengthy tutorial.

If you call to transfer program files, enter a "J" at the main menu prompt to jump to CP/M. Once under CP/M control, execute the MAP transient program to get a system map, which will guide you around the system.

Rather than spend a lot of time talking here about what you can expect when calling my RCP/M system, I suggest that you pick up your phone and call. Again, please leave a message outlining your impressions of the system and any improvements that you might want to suggest.

The Free Software Catalog and Directory

Public domain software is software that legally can be copied, used, and given to another user without payment of a royalty to its author.

Estimates project that over 40 Mb of program source code are contained on the 300+ volumes of CP/M public domain software. In this vast library can be found just about any type of utility program that you might need to help manage your CP/M system. A number of business application programs, mostly written in CBASIC, are present as well.

Until now, many would-be public domain software freaks have been defeated by the thought of searching through over 5000 program titles to find just the few desired. Fortunately, Robert A. Froehlich, a long-term member of SIG/M and an avid user of public domain software, foresaw this problem and has compiled a reference book detailing and indexing each program.

The Free Software Catalog and Directory is a 475-page 8½ x 11-inch book printed on newsprint stock. It focuses on the two largest and most readily available free software libraries, the CP/M Users Group (CPMUG) and the Special Interest for Microcomputers (SIG/M), while providing all the information necessary for personal computer owners to take advantage of this resource.

The introduction begins by setting aside any fears one might have about using public domain software. It concludes by talking a little about the main sources of free software. But most important is the elaborate tutorial—the best I have seen—for successfully completing a session with a remote bulletin board system (BBS). This section in itself makes the book a worthwhile investment for the novice and experienced user alike. The remainder of the introduction prepares the reader for dealing with source code files and explains how to go about making changes to them, should that be necessary.

The real meat of this book is contained in the next six sections. Every file in the entire 92-volume set of the CPMUG library and the first 162 volumes of the SIG/M library is described in detail. The first and second sections, the largest of the six, include complete information for each file as follows: filename, size in Kbytes, CRC checksum, date of entry, language if program source code, author's name, revisor's name, file title, keywords assigned to the file, and finally a textual description of the file.

Of course, the main purpose of this book is to provide ready access to information about the routines or programs of interest to you. The next four sections cross reference each file alphabetically by keyword, language type, author name, and filename. For example, if you want to find a program to compare two files for equality, your initial search might start in the keyword section under utilities. You find there two entries. If the one

Not long ago, *PC* Magazine called MDBS III "The most complete and flexible data base management system available for microcomputers." That's a powerful statement. But then, MDBS III is an amazingly powerful software package. So powerful, in fact, that it lets you build mainframe-quality application systems on your micro or mini. MDBS III is not for beginners. It's for application developers with large data bases or complex data interrelationships who want to define data base structures in the most natural way—without resorting to redundancy or artificial constructs. It's for professionals who can appreciate its extensive data security and integrity features, transaction logging, ad hoc query and report writing capability and its ability to serve multiple simultaneous users. And if you want the power and the glory that only the world's most advanced data management system can provide, MDBS III is for you. For information on MDBS III and our professional consulting services, write or call Micro Data Base Systems, Inc., MDBS/Application Development Products, 85 West Algonquin Road, Suite 400, Arlington Heights, IL 60005. (800) 323-3629, or (312) 981-9200. **MDBS III. ABSOLUTE POWER.**

WE'LL GIVE YOU THE POWER.

YOU TAKE THE GLORY.

MDBS III is a trademark of Micro Data Base Systems, Inc.

$5.00 C Compiler

Due to popular demand, **Dr. Dobb's Journal** has reprinted its most-asked-for C compiler articles by Ron Cain and J. E. Hendrix, each for only $5.00.

Ron Cain's C compiler from sold-out 1980 issues #45 and #48 includes "A Small C Compiler for the 8080s" and "Runtime Library for the Small C Compiler."

The J. E. Hendrix reprint includes part two of "Small-C Compiler v.2" from sold out issue #75 and completes the first part of the compiler article from issue #74 which is included in Dr. Dobb's Bound Volume 7.

To Order: Enclose $5.00 for each copy with this coupon and send to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

Outside U.S., add $2.00 per copy for shipping and handling.

Please send
____ copy(ies) of the Ron Cain reprint, and
____ copy(ies) of the J. E. Hendrix reprint to:

_____
name

_____
address

_____
city state zip

ALL REPRINT ORDERS MUST BE PREPAID.
Please allow 6-9 weeks for delivery.

106

Now Supports Microsoft 3.0

$149.95

Vitamin C

Windows that open, close, grow, shrink, move & scroll. Input validation,formatting, editing & processing. Help messages by field, by key word & from help file. Date & time math, attribute control, pull down menus, and more!

Now even your simplest applications can easily include features that you didn't used to have time or patience enough to tackle. Vitamin C is more than a library full of building blocks. It is a well planned, tightly woven set of high level functions for quick results PLUS low level routines for complete control. Complete source code, no royalties. Great manual with tutorial and reference. Sample programs too! For Microsoft 3.0, CI-C86, & Lattice C. Call for other versions, systems & products.

CALL TODAY ! Or send 149.95+ $3 shipping and handling. Texas add sales tax. MasterCard and Visa accepted.

Creative Programming
Box 112097
Carrollton, Tx 75011
(214)783-9388
Extended Technical support availab le.

Circle **no. 32** on reader service card.

Announcing . . .

Dr. Dobb's Journal Bound Volume 7

Includes every 1982 issue of Dr. Dobb's Journal

Yes! Please send me Volume #7
I enclose ☐ check/money order.
Please charge my: ☐ VISA ☐ M/C ☐ American Express

Card # _____ Expiration Date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Volume 7 _____ x $30.75 = $ _____

Payment must accompany your order. Mail to Dr. Dobb's Journal.
2464 Embarcadero Way, Palo Alto, CA 94303.
Allow 6-9 weeks for delivery.

Postage & Handling must be included with your order.
Please add $1.25 per book in U.S., $2.00 each outside U.S.

☐ Please send me more information on other Bound Volumes.

106

for comparing binary files sounds like it would serve well, note the reference: SIG/M volume 115. You can then track down a complete description of the chosen file by referring to the second section of the book, which covers the SIG/M volumes.

As if all this weren't enough, the book concludes by listing several hundred user groups and bulletin board systems by name, location, and phone number.

This book has become the most used of any in my bookcase. It is a tremendous bargain at $9.95. The author has accomplished a phenomenal task in both compiling this much material and cross indexing it for ease of access. Whether you are an individual user of free software or the librarian of a user group, you will benefit from having this book.

If you're having trouble finding a source of free software in your local area, call the "CP/M Exchange" RCP/M system at (404) 449-6588 for a list of user groups where public domain software can be found.

8088: 8- or 16-bit Processor?

The following paragraphs, taken from the introduction to the Intel *iAPX 88 Book*, set the record straight on how the manufacturer intended the 8088 to be classified:

"This book describes the unique Intel 8088 microprocessor, the outstanding choice for 8-bit microcomputer applications requiring both high performance and low cost.

"The Intel 8088 is the most powerful 8-bit microprocessor available today, yet as easy to use as other 8-bit microprocessors designers have used for years."

Outside of the marketing rhetoric, these two paragraphs contain the phrase "8-bit" often enough to dispel any doubt that the 8088 was designed as an 8-bit processor.

Exiting Properly to CP/M

Two accepted methods for returning to CP/M from an application program are documented by DRI in the CP/M 2.2 reference manual. Although both are said to achieve the same result, in practice they react very differently and, depending on the situation, can cause a great deal of frustration.

Steve Russell of SLR Systems brought this incompatibility to my attention. In the process of testing one of his assemblers, he stumbled onto a difference between exiting to CP/M by jumping to memory location zero (BOOT) and by calling the BDOS with a System Reset function. The problem occurred when he tried executing several assemblies through a submit file in conjunction with XSUB. The first assembly finished successfully, but all the others failed because for some reason XSUB was no longer active after the first assembly.

Apparently, the documentation wasn't updated when XSUB was released. When loaded, XSUB's first action is to relocate itself immediately under the CCP and to modify the BOOT address so that it no longer points to the BIOS jump table but to the XSUB warm start routines. This modification is necessary so that XSUB can trap any further warm start requests to avoid having to reload memory with a fresh copy of the operating software. Thus, XSUB remains resident and active. But if the BDOS is called with the System Reset function, a fresh copy of the operating software is immediately loaded without checking whether XSUB or a similar program is active.

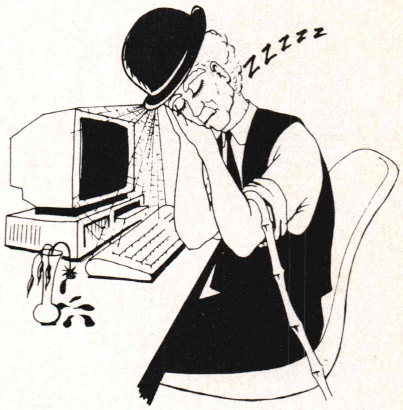
Because XSUB was designed for use within SUBMIT batch procedures, the result of a program returning to CP/M via the BDOS reset call is that suddenly XSUB no longer remains active: the BOOT vector stored at memory location 0 has been reset, and a fresh copy of the operating software is loaded. Any other programs in the same procedure that depend on XSUB to provide console input from the batch file will then fail to execute as planned.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 197.**

BORED?...

...waiting for C programs to compile and link?



Use **C-terp**
the **complete** C interpreter

This is the product you've been waiting (and waiting) for!

Increase your productivity and avoid agonizing waits. Get instant feedback of your C programs for debugging and rapid prototyping. Then use your compiler for what it does best...compiling efficient code ...slowly.

C-terp Features

- Full K&R C (no compromises)
- Complete built-in screen editor-- no half-way house, this editor has everything you need such as multi-files, inter-file move and copy, global searching, auto-indent, tab control, and much more.
- Fast--Linking and semi-compilation are breath-takingly fast. (From edit to run completion in a fraction of a second for small programs.)
- Convenient--Compiling and running are only a key-stroke or two away. Errors direct you back to the editor with the cursor set to the trouble spot.
- Object Module Support -- Access functions and externals in object modules produced by C86 or Lattice C or assembly language. Utilize your existing libraries unchanged!
- Complete Multiple Module Support-- Instant global searches, auto-compile everything that's changed, etc.
- Many more features including batch mode, 8087 support and symbolic debugging.
- Runs on IBM PC, DOS 2.x, 192K and up.
- **Price: $300.00 (Demo $45.00) MC, VISA**

Price of demo includes documentation and shipping within U.S. PA residents add 6% sales tax. Specify C86 or Lattice version.

GIMPEL SOFTWARE
3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: C86 *(Computer Innovations)*, Lattice *(Lattice Inc.)*, IBM *(IBM Corp.)*, C-terp *(Gimpel Software)*

YOU NEED A GOOD LIBRARY

COMPLETE SOURCES NO ROYALTIES

COMPREHENSIVE C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."
— Dr. Dobb's Journal, July 1984

USEFUL "...can be used as an excellent learning tool for beginning C Programmers..."
— PC User's Group of Colorado, Jan. 1985

FLEXIBLE Most Compilers and all Memory Models supported.

RECOMMENDED "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time." — Microsystems, Oct. 1984

- **PACK 1: Building Blocks I** $149
  DOS, Keyboard, File, Printer, Video, Async
- **PACK 2: Database** $399
  B-Tree, Virtual Memory, Lists, Variable Records
- **PACK 3: Communications** $149
  Smartmodem™, Xon/Xoff, X-Modem, Modem-7
- **PACK 4: Building Blocks II** $149
  Dates, Textwindows, Menus, Data Compression, Graphics
- **PACK 5: Mathematics I** $99
  Log, Trig, Random, Std Deviation
- **PACK 6: Utilities I (EXE files)** $99
  Arc, Diff, Replace, Scan, Wipe

Master Card/Visa, $7 Shipping, Mass. Sales Tax 5%

ASK FOR FREE DEMO DISKETTE

NOVUM ORGANUM INC.

SOFTWARE HORIZONS INC.

165 Bedford St., Burlington, MA 01803
(617) 273-4711

Circle **no. 90** on reader service card.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.
4	AGS Computers, Inc.	C-2
2	Addison-Wesley Publishing Co.	64
*	Alcor Systems/MIX Software	15
3	Amanuensis, Inc.	82
16	Arity Corporation	123
51	Artisoft	33
5	Ashton Tate	25
7	Austin Code Works	95
12	BD Software, Inc.	22
8	Blaise Computing	100
14	Borland International	C-4
1	Boston Software Works	83
11	C Journal	26
19	C Source	86
17	C User's Group	113
57	C Ware	103
18	C Ware	34
24	Cogitate, Inc.	83
38	Command Software Systems	23
21	Computer Faire	73
22	Computer Helper Industries, Inc.	87
96	Computer Innovations	67
32	Creative Programming Consultants	126
28	D&W Digital	121
*	*DDJ* Classified Advertising	89
31	Data Base Decisions	105
29	Datalight	14
33	Digital Research Computers	39
35	Ecosoft, Inc.	101
13	Edward K. Ream	118
52	Entelekon Software Systems	102
36	Essential Software	117
30	Everest Solutions	C-3
37	Fair-com	93
40	Fox Software, Inc.	119
*	Gimpel Software	127
*	Gimpel Software	98
45	Golemics	27
43	Greenleaf Software, Inc.	63
26	Hallock Systems Consultants	18
44	Harvard Softworks	16
53	Hayden Book Company	29
48	HiSoft	79
68	IBM Corporation	12-13
46	Illyes Systems	17
*	Integral Quality	122
23	Integral Systems	18
15	Integrand Research Corp.	54
*	JDR Microdevices	69
55	Laboratory Microsystems Inc.	40
104	Lahey Computer Systems	111
36	Lattice, Inc.	36
125	Lifeboat Associates	28
56	Living Software	1
9	Logique	37
6	Lugaru Software, Ltd.	44
122	Micro Data Base Systems, Inc.	125
54	MacTutor	36
62	Manx Software	60-1
84	Megamax, Inc.	87
61	Microcompatibles	70
73	Micro Dynamics	49
134	Micro Software Developers	55
74	Micro Solutions	59
64	Microprocessors Unlimited	79
*	Microrim, Inc.	7-9
41	MicroSmith	18
60	Microtec	2
66	Mitec	74
128	Morgan Computing Company	111
83	Motel Computers Ltd.	113
79	Mystic Canyon Software	21
87	O.E.S. Systems	79
39	Overland Data, Inc.	120
76	Personal Tex Inc.	17
69	Plu Perfect Systems	78
70	Plum Hall, Inc.	31
71	Poor Persons Software	85
67	Programmer's Connection	65
97	Programmer's Shop	53
72	Programmer's Shop	51
25	Que Corporation	105
91	Raima Corporation	97
49	Relational Database Systems	57
80	Revasco	37
78	SLR Systems	121
47	STS Enterprises	27
114	Seidl Computer Engineering	41
85	SemiDisk Systems	45
86	Shaw American Technologies	41
124	Social & Scientific Systems	41
63	Soft Advances	121
88	Softaid, Inc.	109
*	Softfocus	14
90	Software Horizons, Inc.	128
99	Software Toolworks	71
92	Softway, Inc.	35
93	Solution Systems	58
94	Solution Systems	58
95	Solution Systems	58
75	Solution Systems	58
102	Solution Systems	81
118	Solutionware	111
59	Speedware	49
65	Spruce Technologies Corp.	3
10	Sunny Hill Software	113
50	Systems Guild	118
100	Thunder Software	109
20	Trio Systems	75
81	Turbo Power Software	43
77	UniPress Software	19
106	User's Guide	77
27	Vermont Creative Software	99
120	Versasoft Corporation	46-7
112	Wendin, Inc.	11
130	Western Wares	109
108	Whitesmiths, Ltd.	62
116	Wizard Systems	85
110	Zebra Systems	83
*	*DDJ* Bound Volume	126
*	*DDJ* Subscription Problem	27
*	*DDJ* Compiler	126

* This advertiser prefers to be contacted directly: see ad for phone number.

Advertising Sales Offices

East Coast
Walter Andrzejewski (617) 567-8361

Midwest/West Central
Michele Beaty (317) 875-0557

Northern California/Northwest
Lisa Boudreau (415) 424-0600

Southern California/Southwest
Beth Dudas (714) 643-9439

Classified Advertising
Tim Ortiz (415) 424-0600

Advertising Director
Shawn Horst (415) 424-0600

Advertising Coordinators
Ronald Copeland (415) 424-0600
Alice Abrams (415) 424-0600

H.E.L.P. Eliminates Every Bug known to Compilers . . . As well as a few other species

H.E.L.P. is a completely interactive C programming environment with three innovative full-sized features that will revolutionize the way you write code.

A Clean Compile — Guaranteed!

Say Good-bye to all compiler-type errors. **H.E.L.P.** 's built-in program checker (which would embarrass LINT) not only hunts down bugs . . . explains the proper syntax . . . gives examples of usage . . . but will even offer suggested corrections. If you want, **H.E.L.P.** will even make the corrections for you . . . at the touch of a key.

H.E.L.P. also finds semantic errors as well as poor style and inefficiencies. You can even check the portability of your code!

Multi-Window Editing

Open as many windows as you want . . . . there's no limitation . . . not even your own memory. Because **H.E.L.P.** uses a virtual memory system you can create programs larger than your machine capacity.

H.E.L.P.'s very powerful editor allows you the flexibility to work in several windows . . . with several files at the same time.

Save Keystrokes

Hundreds of commands are bound to the keyboard to give you fast execution.

Always be in Control

Not only can you develop code in many windows at the same time, but you can show (and refer to) important definitions in one window while creating in another. Or open a window and keep notes about your program . . . or type a memo . . . or a letter.

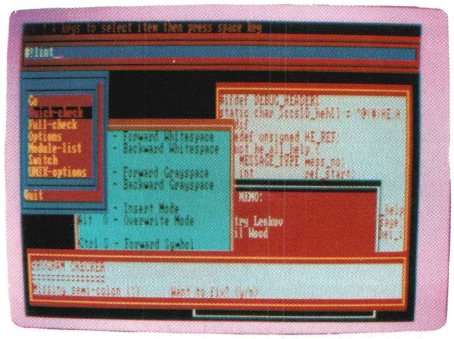
Increase your Productivity by 300% or More . . . . .

If you are a novice programmer, you'll begin writing code like an advanced programmer much faster with **H.E.L.P.**

Just imagine what **H.E.L.P.** will do for the ADVANCED PROGRAMMER.

You'll have more time to become creative with your algorithm (since **H.E.L.P.** will make sure your code compiles the "first time at bat").

H.E.L.P. tracks every step you make. If you are not sure about a command, just press a key, and you'll get the kind of help you need.



Check These Features

- *Multi-window environment*
- *Interactive program checking*
- *Check syntax, semantic, type usage, intermodule inconsistencies and portability*
- *Multi-file editing*
- *Intelligent help subsystem*
- *User-definable keyboard bindings*
- *Supports color and monochrome*
- **H.E.L.P.** *supports the full C Language*

NOW IN MS-DOS

EVEREST SOLUTIONS

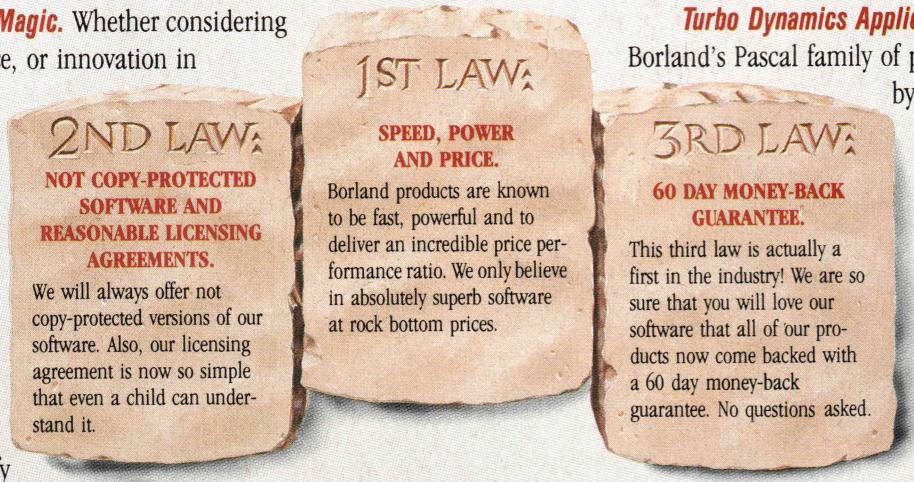
Order now $395

Everest Solutions, Inc.
3350 Scott Boulevard
Building 58
Santa Clara, CA 95051
(408) 986-8977

PC WEEK'S PRODUCT OF THE YEAR
PC MAGAZINE'S AWARD FOR TECHNICAL EXCELLENCE

Borland Introduces the Laws of *TURBO DYNAMICS*™

Laws That Work Like Magic. Whether considering technological excellence, or innovation in areas such as pricing, not copy-protection, licensing agreements, site licenses, 60 day money-back guarantee —Borland is clearly recognized as the software industry leader. The following three laws of "*Turbo Dynamics*"™ exemplify our pledge for excellence.



Turbo Dynamics Applies to Turbo Pascal. Borland's Pascal family of products is growing by leaps and bounds. You can now join hundreds of thousands of users and enter the world of Turbo Pascal programming. And remember, all three laws of *Turbo Dynamics* apply to all Borland products.

TURBO PASCAL™ $69.95

The industry standard. With more than 350,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use. **Free spreadsheet** included on every Turbo disk with ready-to-compile source code. ***Options:*** We offer the exciting Binary Coded Decimal (BCD) option for your business applications as well as an 8087 option for your number-crunching applications at a very low charge. Please refer to the coupon. ***Portability.*** Turbo Pascal is available today for most computers running PC-DOS, MS-DOS, CP/M-80 or CP/M-86. ***Jeff Duntemann, PC Magazine:*** "In its simplicity it achieves an elegance that no other language compiler has ever displayed."

TURBO GRAPHIX TOOLBOX™ $54.95

High resolution monochrome graphics for the IBM PC.
The Turbo Graphix Toolbox will give even a beginning programmer the expert's edge. It's a complete library of Pascal procedures and functions. Tools that will allow you to draw and hatch pie charts, bar charts, circles, rectangles and a full range of geometric shapes. Procedures that will save and restore graphic images to and from disk. And much, much, more. You may incorporate part or all of these tools in your programs and yet we won't charge you any royalties. Best of all, these functions and procedures come complete with commented source code on disk ready to compile.

TURBO TUTOR™ $34.95

From start to finish in 300 pages. Turbo Tutor is for everyone from novice to expert. Even if you've never programmed before Turbo Tutor will get you started right away. ***A must.*** You'll find the source code for all the examples in the book on the accompanying disk ready to compile. Turbo Tutor might be the only reference on Pascal and programming you'll ever need.

TURBO DATABASE TOOLBOX™ $54.95

The Turbo Database Toolbox is the perfect complement to Turbo Pascal. It contains a complete library of Pascal procedures that allows you to sort and search your data and build powerful applications. It's another Borland set of tools that will give the beginning programmer the expert's edge. ***Get started right away: free database!*** Included on every Toolbox disk is the source code to a working data base which demonstrates how powerful and easy to use our search system, Turbo-Access, really is. Modify it to suit your individual needs or just compile it and run. ***Remember, no royalties!***

NOT COPY-PROTECTED

TURBO PASCAL FAMILY

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

Pascal 3.0	$ 69.95
Pascal w/8087	$109.90
Pascal w/BCD	$109.90
Pascal w/8087 & BCD	$124.95
Turbo Database Toolbox	$ 54.95
Turbo Graphix	$ 54.95
Turbo Tutor	$ 34.95

*These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

60 DAY MONEY-BACK GUARANTEE

Carefully Describe your Computer System!

Mine is: ___ 8 bit ___ 16 bit
I Use: ___ PC-DOS ___ MS-DOS
___ CP/M 80 ___ CP/M 86

My computers' name/model is: _____

The disk size I use is:
☐ 3½" ☐ 5¼" ☐ 8"

Name: _____
Shipping Address: _____
City: _____
State: _____ Zip: _____
Telephone: _____

Amount: (CA 6% tax) _____
Payment: VISA MC BankDraft Check
Credit Card Expir. Date: ___/___
Card #: _____
11

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax. Outside USA: add $10 and make payment by bank draft, payable in US dollars drawn on a US bank.

BORLAND INTERNATIONAL
4585 Scotts Valley Drive, Scotts Valley CA 95066
Phone (408) 438-8400 Telex 172373

Copyright 1985 Borland International BI-1011

Turbo Pascal, Turbo Database Toolbox, Turbo Graphix Toolbox, Turbo Tutor and Turbo Dynamics are trademarks of Borland International, Inc.

Circle no. 14 on reader service card.